VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ TVÁŘÍ

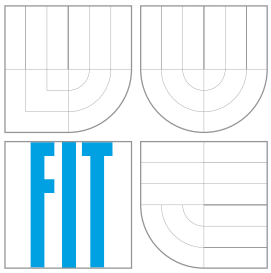
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL DUBAN

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ TVÁŘÍ

FACE RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL DUBAN
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. PAVEL SMRŽ, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Hlavním cílem této bakalářské práce bylo vytvořit aplikaci spustitelnou na robotu PR2, která je schopna automaticky detekovat a učit se různé lidské tváře, které se postupně objevují na videu pořizovaném z kamer umístěných na tomto robotovi.

Abstract

This Bachelor's thesis aimed to develop an application, executable by the robot PR2, capable of automatical recognizing and memorizing human faces which are displayed successively on video recorded by cameras placed on the robot.

Klíčová slova

počítačové vidění, OpenCV, detekce tváří, rozpoznávání tváří, zpracování obrazu, robotika, Robot Operating System (ROS), robot PR2

Keywords

computer vision, OpenCV, face detection, face recognition, image processing, roborics, Robot Operating System (ROS), robot PR2

Citace

Michal Duban: Rozpoznávání tváří, bakalářská práce, Brno, FIT VUT v Brně, 2014

Rozpoznávání tváří

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Duban
21. května 2014

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu panu doc. RNDr. Pavlu Smržovi, Ph.D. za ochotu a odbornou asistenci, kterou mi při řešení tohoto projektu poskytl. Dále bych pak chtěl poděkovat panu Ing. Michalu Španělovi, Ph.D. za pomoc při manipulaci s robotem PR2.

Obsah

Kapitola 1

Úvod

Obor počítačového vidění je v informatice relativně krátkou dobu. Za jeho vznik jsou považována šedesátá léta dvacátého století, kdy byl nalezen způsob, jak získat 3D geometrické informace z 2D obrazu [6]. Od této doby se počítačové vidění postupně zdokonalovalo a rozvíjelo. Přispěla k tomu především větší dostupnost stále výkonnějších počítačů. Toto umožňovalo zpracovávat čím dál větší objemy dat vyšší rychlostí. Postupně vznikaly nejrůznější knihovny a frameworky, které uživatelům práci s obrazem ještě více zjednodušovaly a zpřístupňovaly.

Počítačové vidění postupně začalo pronikat do různých dalších vědních oborů, kde se snažilo zjednodušit lidem jejich práci. Nejdůležitějšími disciplínami, kde počítačové vidění našlo své uplatnění je fyzika, neurobiologie a robotika [11]. V dnešní době však můžeme narazit na prostředky podporující počítačové vidění téměř kdekoliv a do kontaktu s nimi se dostává prakticky každý. Příkladem může být fotoaparát mobilního telefonu, který nám během focení detekuje a označí lidskou tvář a podle toho daný algoritmus sám zaostří a pořídí snímek.

Hlavním cílem této bakalářské práce je tedy propojit počítačové vidění s robotikou. Výsledkem projektu je aplikace, která je spustitelná na robotovi PR2 a která je schopna detekovat a naučit se různé lidské tváře, jež jsou zachyceny kamerou z tohoto robota.

Ve druhé kapitole pojednávám o knihovně OpenCV, kterou jsem použil pro práci s obrazem. Rovněž zde popisuji základní principy a výhody zvolených algoritmů pro samotnou detekci a rozpoznávání lidských tváří v obraze.

Třetí kapitola obsahuje stručný popis robota PR2 a frameworku ROS, který usnadňuje tvorbu a vývoj robotických aplikací a samotnou komunikaci s robotem.

Ve čtvrté kapitole jsou nastíněny jednotlivé postupy a algoritmy, které jsem sám navrhl a implementoval. Je zde popsána koncepce celé aplikace a její ovládání. Dále jsou zde uvedeny jednotlivé parametry, kterými lze ovlivnit funkčnost a chod programu.

Pátá kapitola je věnována popisu jednotlivých testů, kterými jsem ověřoval správnou funkčnost výsledné aplikace. V testech jsou zohledněny různé faktory, které mohou snížit úspěšnost správného rozpoznání či detekce naučené osoby.

Závěrečná kapitola obsahuje shrnutí a zhodnocení dosažených výsledků. Jsou zde zmíněny i nápady na případné vylepšení nebo rozšíření aplikace.

Kapitola 2

Základní pojmy počítačového vidění

Tato kapitola slouží jako teoretický úvod do problematiky zpracování obrazu. Nalezneme zde vysvětlení základních principů počítačového vidění a popis jednotlivých algoritmů pro detekci a rozpoznávání lidských tváří, které poskytuje knihovna OpenCV a kterých ve své bakalářské práci využívám.

2.1 Počítačové vidění

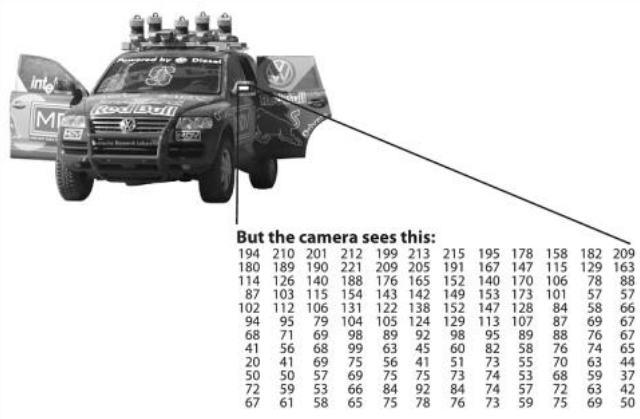
Počítačové vidění lze definovat několika způsoby. Jednou z mnoha definic je například definice z knihy [7].

Definice 1 *Počítačové vidění je způsob využití výpočetní techniky pro extrakci, charakteristiku a interpretaci vizuálního obrazu trojrozměrného světa.*

Počítačové vidění je vědní obor, který se zabývá zpracováním obrazu reálného světa. I v dnešní době však lze tuto disciplínu považovat za velmi náročnou a složitou. Počítačové vidění dokáže mnohdy řešit jen omezené množství úloh, které musí být často do značné míry zjednodušeny, abychom dospěli k rozumným výsledkům.

Celkový postup počínající snímáním videa a končící porozumění samotnému obsahu nasnímaných dat lze popsat pomocí posloupnosti následujících dílčích kroků:
Dělení a popis jednotlivých kroků bylo převzato z následujících publikací [15, 5].

1. **Snímání, digitalizace a uložení obrazu v počítači.**
   Abychom byli schopni s obrazem pracovat, musí existovat zařízení, prostřednictvím kterého informace o okolním světě vstupují do počítače. Toto ve většině případů zajišťují různé typy kamer, které převádějí vstupní optické veličiny na elektrické signály. Mezi typické zdroje vstupních signálů patří ultrazvuk, jas nebo tepelné záření.
   Takto získané spojité signály jsou dále digitalizací převedeny do diskrétního tvaru. Vstupní signál je tedy vzorkován a kvantován. Výsledkem těchto procesů je matice pixelů, která popisuje získaný obraz. Pixely jsou tedy základní, dále nedělitelnou jednotkou obrazu a jsou reprezentovány přirozenými čísly. Reprezentace obrazu v jednotlivých pixelech je zobrazen na obrázku 2.1.



Obrázek 2.1: Reprezentace obrazu pomocí jednotlivých pixelů[1].

Obrázek 2.2: Obrázek před a po aplikaci algoritmu na detekci hran.[2]

2. **Předzpracování.**
   Tento krok lze považovat za tzv. nižší úroveň zpracování obrazu. V této úrovni nás nezajímá sémantika zpracovávaných dat. Snažíme se naopak o potlačení šumu a zkreslení, které mohlo vzniknout při přenosu obrazu a při jeho digitalizaci. Dalším úkolem této fáze zpracování obrazu je nalezení symbolických informací, které jsou dále zpracovávány v tzv. vyšší úrovni. V závislosti na úloze může docházet k aplikaci různých filtrů a transformací, které mohou například zvýraznit určité rysy nebo hrany, což může vést k lepší interpretaci rozpoznávaných dat v dalších krocích.

3. **Segmentace obrazu na objekty.**
   V tomto kroku se snažíme do jisté míry porozumět a vyhodnotit to, co se v obraze nachází. Snažíme se v něm tedy najít objekty různých tvarů a velikostí. Tento krok je považován za jeden z nejnáročnějších kroků a k jeho realizaci je mnohdy využívána umělá inteligence.

4. **Popis objektů.**
   Po nalezení jednotlivých objektů je nutné provést jejich klasifikaci a popis. Popisování se provádí buďto na základě souboru číselných charakteristik, nebo na základě vztahů mezi jednotlivými objekty.

[1] Obrázek byl převzat z knihy Learning OpenCV [2]
[2] Obrázek byl převzat ze stránek http://www.iplt.org/wiki/wiki/AlgSobel

5. **Porozumění obsahu obrazu (často jen klasifikace objektů).**
   Jedná se například o klasifikaci objektů do různých tříd, které se mohou lišit kupříkladu tvarem.

2.2 Knihovna OpenCV

OpenCV je svobodná multiplatformní knihovna, která se zabývá zpracováním obrazu v reálném čase. Uživatelům nabízí velké množství již implementovaných a odladěných algoritmů, které usnadňují a zpřístupňují oblast počítačového vidění i běžným uživatelům. Knihovnu lze využívat z programovacích jazyků C, C++ nebo Python [2]. Při popisu jednotlivých algoritmů detekce a rozpoznávání jsem vycházel z následujících publikací [9, 8, 2, 16, 1, 10, 3].

Také já jsem ve své bakalářské práci využil vybrané algoritmy, které tato knihovna nabízí a kterým bych se v této části rád věnoval. Jedná se především o algoritmy sloužící na detekci a rozpoznávání lidských tváří.

2.2.1 Detekce lidských tváří

Hlavním úkolem detektoru lidských tváří je lokalizovat lidský obličej v obraze. Typickým vstupem takovéhoto detektoru je obrázek pořízený například z videokamery nebo fotoaparátu. Výstupem detektoru je pak pozice tváře, tedy souřadnice, na kterých se detekovaný obličej nachází. V dnešní době jsou takovéto detektory velmi rozšířené a lze je nalézt v téměř každém mobilním telefonu nebo fotoaparátu. Knihovna OpenCV poskytuje pro detekci tváří algoritmus Haarových kaskád (Haar cascades), který ve své práci využívám.

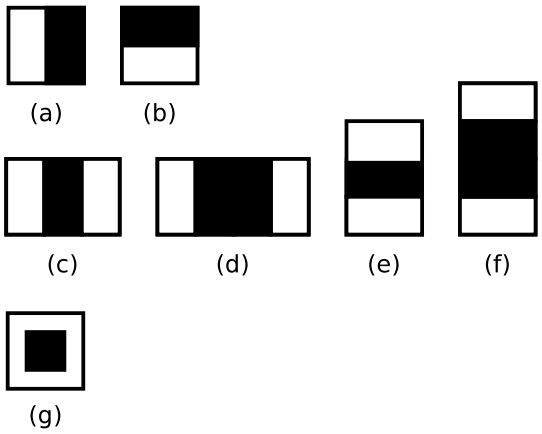
Tento algoritmus je založen na Haarových příznacích, což je algoritmus sloužící k detekci různých objektů v obraze. Algoritmus byl poprvé představen v roce 2001 a jeho zakladateli jsou P. Viola a M. Jones. Pro jeho správnou funkci je potřeba velké množství pozitivních a negativních vzorků daného objektu, na kterých je tento algoritmus trénován. K trénování detektoru je využito algoritmu AdaBoost, který obsahuje kaskádu jednoduchých klasifikátorů, která je vytvořena na základě trénovací množiny snímků [9]. Velkou výhodou tohoto algoritmu je, že si sami můžeme sestavovat trénovací množiny a tedy naučit detektor rozpoznávat nejrůznější předměty a objekty. Pro účely tohoto projektu jsem použil již předpřipravený klasifikátor `haarcascade_frontalface_alt.xml`, který nabízí knihovna OpenCV.

Princip algoritmu Haarových kaskád

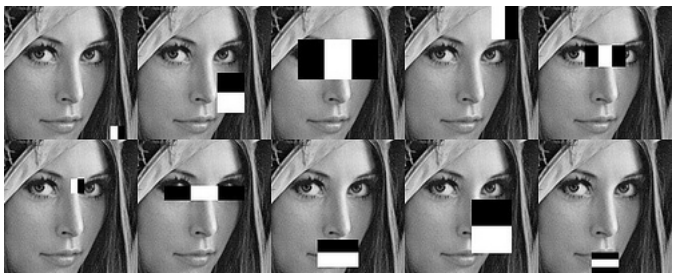
Pro detekci hledaného objektu v obraze je nejprve potřeba zjistit hodnoty jednotlivých Haarových příznaků. Tyto příznaky jsou odvozeny z několika obdélníků, které se dělí podle toho, jaký typ informace má být podle nich v obraze nalezen. Na obrázku 2.3 můžeme vidět různé typy Haarových příznaků. Jak je na obrázku 2.3 patrné, příznaky se skládají z obdélníků bílé a černé barvy. Každá tato barva má stanovenou patřičnou váhu. Váha bílé oblasti je stanovena na $w_0 = -1$. Váha černé oblasti je vypočtena jako podíl ploch bílé a černé oblasti. Hodnota Haarova příznaku je pak rovna odezvě obrazu, na který byl daný příznak aplikován. Jeho konkrétní hodnota je dána následujícím vztahem:

$$f(x) = w_0 r_0 + w_1 r_1 \tag{2.1}$$

kde $f(x)$ je odezva Haarova příznaku na vstupní snímek x, w_0 je váha bílé obdélníkové oblasti r_0, w_1 je váha černé obdélníkové oblasti r_1. Daný příznak je následně iterativně



Obrázek 2.3: Obrázky a) a b) znázorňují hranové Haarovy příznaky c), d), e) a f) zobrazují čárové hranové Haarovy příznaky a obrázek g) reprezentuje středový Haarův příznak.

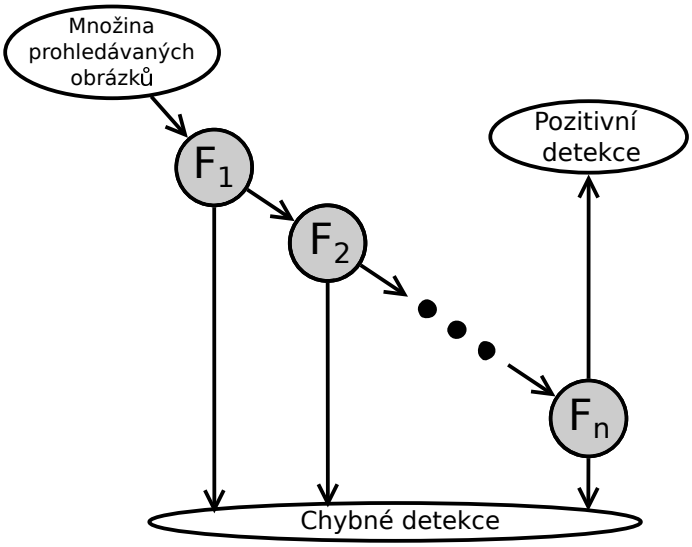


Obrázek 2.4: Aplikace jednotlivých Haarových příznaků na vstupní obraz.[3]

posouván po jednom pixelu v horizontálním i vertikálním směru. Pokud je příznak posunut přes celé okno, dojde k jeho zvětšení a celý proces se opakuje. Tento postup opakujeme do té doby, než velikost příznaku je větší než velikost okna [8]. Na obrázku 2.4 můžeme vidět aplikaci jednotlivých Haarových příznaků na prohledávaný obraz.

Jednou z optimalizací výpočtu hodnot jednotlivých Haarových příznaků je převedení vstupního obrazu na tzv. Integrální obraz. Touto metodou se stává počítání hodnot příznaků efektivnější a rychlejší. Nyní je každý vstupní snímek předán ke zpracování kaskádě klasifikátorů. Každý prvek této kaskády je vlastně klasifikátor, který obsahuje pravidla pro vyhodnocení daného obrázku. Tato pravidla jsou definována na základě vypočtených Haarových příznaků a příznaků získaných při trénování daného objektu. Pokud snímek těmto pravidlům vyhoví, je poslán do další úrovně kaskády. Pokud však klasifikátor vyhodnotí snímek na základě příznaků jako neplatný, detekce aktuálního snímku končí (hledaný objekt se na snímku nenachází). Projde-li vstupní obrázek celou kaskádou klasifikátorů, hledání daného objektu je považováno za úspěšné [2]. Princip je ilustrován na obrázku 2.5.

[3]Obrázek byl převzat ze stránek http://sirloon.net



Obrázek 2.5: Kaskáda klasifikátorů.

2.2.2 Rozpoznávání lidských tváří

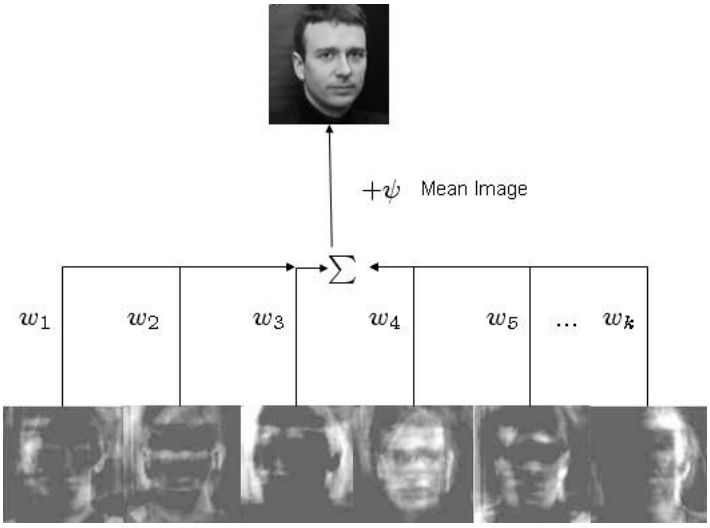
Každá lidská tvář je do jisté míry unikátní a originální i navzdory tomu, že obsahuje určité prvky a oblasti, které jsou pro všechny lidi společné. Těmito prvky jsou například oči, nos, ústa nebo brada (jedná se o tzv. charakteristiky tváře). Aby počítač byl schopen jednotlivé obličeje rozeznat, musí si všímat právě těchto charakteristik a musí mezi nimi hledat určité vztahy a zákonitosti. Většina programů a algoritmů, které se s touto problematikou zabývají, rozpoznává tváře na základě vzdáleností, poměrů a úhlů mezi jednotlivými oblastmi obličeje daného člověka. Existuje však mnoho faktorů, které ztěžují nebo dokonce znemožňují hledání charakteristik obličeje v obraze. Mezi nejtypičtější patří:

- Změna pozice nebo otočení tváře
- Změna účesu nebo výrazu tváře
- Používání brýlí, čepic nebo jiných pokrývek hlavy
- Změna světelných podmínek
- Vliv času (stárnutí)

Knihovna OpenCV poskytuje k rozpoznávání tři základní typy algoritmů. V následující části budou postupně tyto algoritmy představeny a vysvětleny. První dva algoritmy (Eigenfaces a Fisherfaces) budou vysvětleny pouze okrajově. Algoritmu local binary patterns bych se rád věnoval více, jelikož je to algoritmus, který jsem použil ve své aplikaci.

Eigenfaces

Algoritmus Eigenfaces je považován za základní algoritmus k rozpoznávání lidských tváří, který přináší relativně uspokojivé výsledky. Algoritmus je založen na vlastních vektorech, což jsou vektory, které po transformaci nemění svůj směr, ale pouze svou velikost. Eigenfaces obsahuje množinu takovýchto vektorů, přičemž jeden tento vektor reprezentuje právě jeden



Obrázek 2.6: Rekonstrukce konkrétní tváře z jednotlivých Eigenfaces.[4]

obrázek z celkové trénovací množiny obrázků. Množina těchto vlastních vektorů je odvozena na základě procesu označovaného jako PCA (z anglického *principal component analysis*). Cílem tohoto matematického procesu je snížení aktuální dimenze dat s co nejmenší ztrátou informace. Spustíme-li tedy PCA nad konkrétní množinou trénovacích snímků jednotlivých tváří, dostaneme jejich zredukovanou podmnožinu, na základě které se skládají jednotlivé aproximace konkrétních obličejů. Každou tvář tedy můžeme do jisté míry vyjádřit jako součet průměrné tváře [16]. Konkrétní příklad rekonstrukce obličeje z redukované množiny Eigenfaces je ilustrován na obrázku 2.6. Kvalita konkrétní rekonstrukce je do značné míry ovlivněna počtem jednotlivých vektorů.
Mezi hlavní výhody této metody patří:

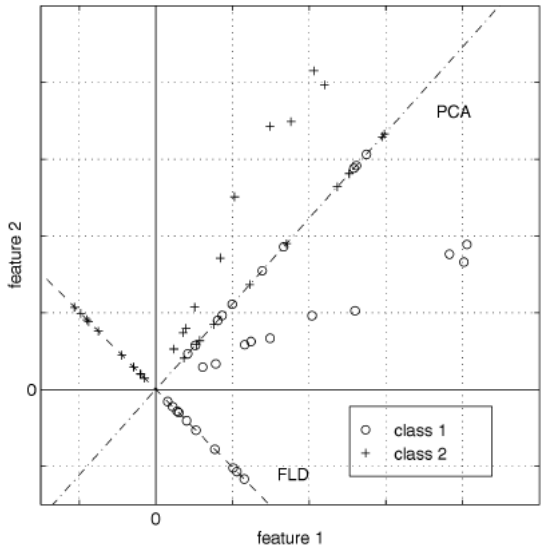
- Rychlost výpočtu
- Jednoduchá implementace
- Možnost práce nad velkou množinou vstupních snímků

Metoda má ale i své nevýhody. Mezi nejzásadnější patří:

- Metodu lze využívat pouze na tváře nacházející se kolmo na kameru
- Špatné výsledky při snímcích pořízených za různých světelných podmínek
- Špatné výsledky v různorodém pozadí

Jelikož je předpoklad, že mnou navržená aplikace bude pracovat se vstupními snímky, které budou pořizovány s pokaždé jiným pozadím (detekovaná tvář se v místnosti může nacházet na rozdílných pozicích) a za různých světelných podmínek (místnost, ze které budou snímky pořizovány, může být pokaždé jinak osvětlena), proto jsem tento algoritmus nezvolil ve své práci jako výchozí algoritmus pro rozpoznávání tváří.

[4]Obrázek byl převzat ze stránek http://onionesquereality.wordpress.com/tag/alex-pentland/



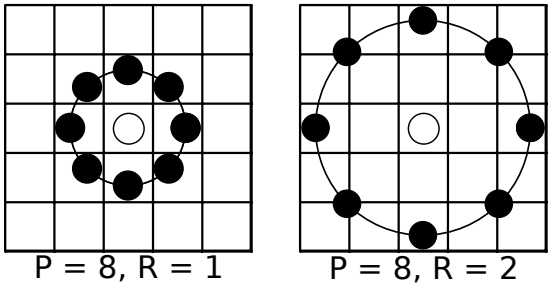
Obrázek 2.7: Srovnání metod používaných algoritmem Eigenfaces (PCA) a Fisherfaces (FLD).[6]

Fisherfaces

Hlavním cílem této metody je odstranění limitujících faktorů, které byly popsány u metody Eigenfaces. Fisherfaces se tedy snaží o to, aby se rozpoznávač dal používat co nejvíce za reálných podmínek, se kterými se v praxi běžně setkáváme. Metoda je zaměřena především na odbourání vlivu světelných podmínek, natočení a gest tváří na rozpoznávání jednotlivých osob. Metoda se dále pokouší minimalizovat případné ztráty informací, které mohou vznikat ve specifických případech při aplikaci algoritmu PCA. Fisherfaces je založena na algoritmu FLD (z anglického *Fisher´s Linear Discriminant*). Tento algoritmus se snaží rozdělit skupiny bodů obrazu do tříd. Z těchto tříd jsou následně vytvořeny dvojice matic. Jeden typ matice reprezentuje maximální rozptyl mezi jednotlivými třídami. Druhý typ matice naopak představuje minimální rozptyl jednotlivých bodů v rámci každé třídy. Následně je z těchto dvou typů tříd vytvořena nová matice, která seskupuje vzory stejných tříd a odděluje vzory tříd různých. Touto metodou tedy nedochází k shlukování specifických bodů, což do značné míry usnadňuje proces rozpoznávání a minimalizuje ztrátu informací. [1] Na obrázku 2.7 je zobrazeno porovnání metod PCA a FLD.

Stejně jako předchozí metoda má i Fisherfaces své nedostatky. Mezi nejzásadnější patří předpoklad, rovnoměrného rozložení dat v jednotlivých třídách. Pokud tento předpoklad není splněn, rozpoznávání selhává. Dále taktéž záleží na vstupní množině snímků, na které se algoritmus učí. Pokud je rozpoznávač naučen pouze na snímcích, které jsou dostatečně osvětleny, je detekce stejných obličejů za špatného osvětlení problematická.

[6] Obrázek byl převzat z publikace Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection [1].



Obrázek 2.8: Příklady různých typů okolí.

Local Binary Patterns

Další metoda poskytovaná knihovnou OpenCV je metoda local binary patterns (LBP). LBP je pro své výhody velice oblíbenou a rozšířenou metodou, která se používá v mnoha aplikacích jak pro detekci objektů tak i pro zpracování textur. Hlavními přednostmi LBP je invariantnost vůči různým světelným podmínkám na vstupních obrazech, výpočetní nenáročnost a rychlost. Z těchto důvodů jsem LBP upřednostnil ve své práci před výše popsanými metodami.

Metoda je založena na prahování okolních pixelů šedotónového vstupního obrazu pomocí pixelu středového. Dochází tedy k postupnému procházení jednotlivých pixelů vstupního obrázku. Z tohoto obrázku je vždy určen jeden hlavní, středový pixel a pixely tento středový pixel obklopující, které definují jeho okolí. Jednotlivá okolí se mohou navzájem lišit počtem a vzdáleností jednotlivých bodů od středového pixelu. Počet bodů je charakterizován parametrem P a vzdálenost od středu je určena hodnotou proměnné R. Konkrétní ukázka jednotlivých typů okolí je viditelná na obrázku 2.8. Souřadnice jednotlivých pixelů okolí $[x_p;y_p]$ jsou dány následujícími vztahy:

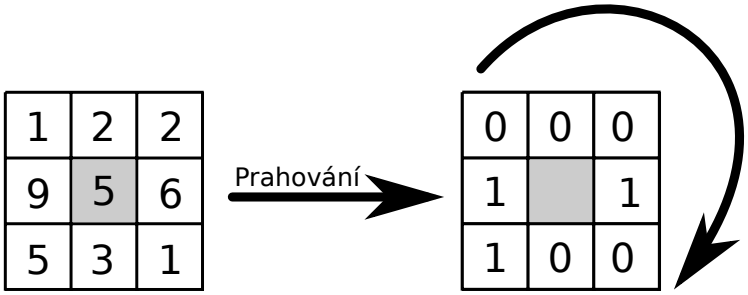
$$\left[x_p = x_c + Rcos(\frac{2\pi p}{P}); \quad y_p = y_c - Rsin(\frac{2\pi p}{P})\right], \tag{2.2}$$

kde x_c a y_c jsou souřadnice středového pixelu a $p \in P$. V případě, kdy vypočtené souřadnice neodpovídají konkrétnímu pixelu, je nová souřadnice získána interpolací [10].

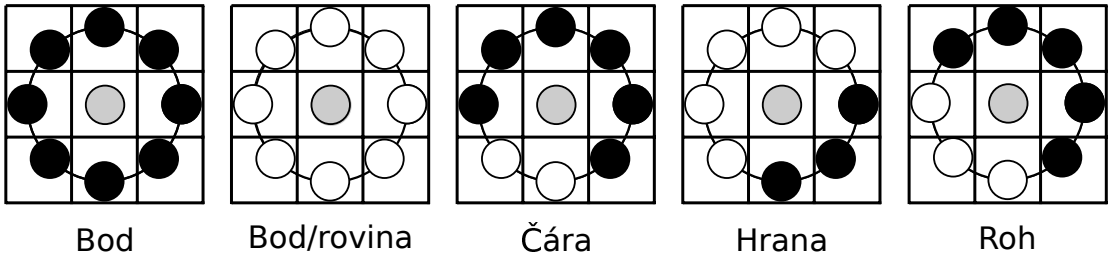
Samotný proces prahování je založen na porovnávání hodnoty středového pixelu s hodnotami pixelů v jeho okolí. Pokud je hodnota středového pixelu větší nebo rovna hodnotě okolního pixelu, je původní hodnota nahrazena 0. V opačném případě je jeho hodnota nahrazena 1. Na obrázku 2.9. je ilustrován postup prahování pro okolí $P = 8$, $R = 1$. Výsledkem prahování je tedy binární číslo charakterizující daný středový pixel. Toto binární číslo lze získat sepsáním jednotlivých bitů okolí. Tyto bity jsou sepisovány ve směru hodinových ručiček, přičemž výchozí pozice je v levém horním rohu. Získané binární číslo je dále převedeno na tzv. LBP kód. Převod se provádí přiřazením binomické váhy každému členu daného okolí. Celý postup výpočtu LBP kódu lze zapsat matematicky následující rovnicí:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c), \tag{2.3}$$

kde i_c je šedotónová hodnota středového pixelu, i_p jsou jednotlivé hodnoty okolních pixelů a 2^p je binomická váha.



Obrázek 2.9: Prahování okolních pixelů pomocí středového pixelu. Výsledkem prahování je binární číslo. V tomto případě tedy 00010011.

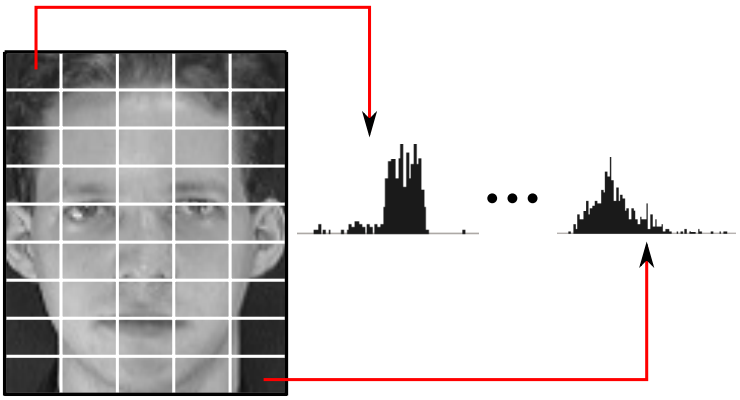


Obrázek 2.10: Reprezentace základních tvarů detekovatelných pomocí LBP. Černý bod reprezentuje 0 a bílý bod reprezentuje 1.

Funkce s je dále definována jako:

$$s(x) = \begin{cases} 1 & \text{pokud } x \geq 0 \\ 0 & \text{pokud } x \leq 0 \end{cases} \tag{2.4}$$

Local binary patterns lze dále využít k detekci základních tvarů v obraze a v texturách. Mezi typické tvary detekovatelné pomocí LBP jsou například roviny, hrany, body nebo různé čáry [10]. Na obrázku 2.10 jsou zobrazena jednotlivá okolí. Je zde taktéž popsán jejich význam.



Obrázek 2.11: Fotografie rozdělena na oblasti. Celkový histogram je dán složením histogramů jednotlivých oblastí.[8]

Pro následné použití LBP pro rozpoznávání jednotlivých tváří je vstupní obrázek obsahující lidský obličej rozdělen na M malých, navzájem se nepřekrývajících oblastí. Tyto oblasti se skládají z předem vypočtených hodnot LBP kódu. Pro každou oblast je nyní vypočten její histogram. Jednotlivé histogramy se postupně skládají za sebe. Tímto postupem je získán celkový histogram, na základě něhož je následně lidská tvář rozpoznávána [3]. Popsaný princip je zachycen na obrázku 2.11.

[8]Fotografie obličeje převzata z databáze obličejů AT&T Facedatabase, která je dostupná na http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Kapitola 3

Platforma ROS a robot PR2

Následující kapitola obsahuje informace o robotu PR2, což je zařízení, pro které je výsledná aplikace navržena a na kterém byla testována. Nalezneme zde základní popis robota a jeho periferií. Dále v kapitole najdeme popis frameworku ROS, který je potřebný pro fungování jednotlivých aplikací na robotu.

3.1 Robot PR2

Robot PR2[1] je zařízení, které je součástí vybavení Výzkumné skupiny robotiky Robo@FIT. Toto zařízení lze do jisté míry považovat za humanoid, jelikož má prvky charakterizující člověka. Robotická ramena, série videokamer nacházejících se po celém robotovi a možnost pohybu celého zařízení dávají robotu možnost orientovat se a pohybovat v lidském prostředí a umožňují mu vnímat a manipulovat s předměty a objekty, které se nacházejí v jeho blízkosti. Díky těmto vlastnostem a rozsáhlé podpoře různých knihoven obsahující spoustu předimplementovaných algoritmů lze relativně snadno pro tohoto robota vytvářet různé aplikace. Jedním z hlavních frameworků ulehčující manipulaci a komunikaci mezi jednotlivými prvky robota je platforma ROS. Více o této platformě je uvedeno v sekci 3.2.

Robot PR2 je komplexní a složité zařízení skládající se z mnoha součástí. V následující sekci budou tyto součásti rozděleny a popsány ve dvou základních podkapitolách. Při popisu vnitřní a vnější struktury robota PR2 jsem vycházel z uživatelského manuálu tohoto robota [12].

3.1.1 Vnější struktura robota PR2

Následující podkapitola se zabývá jednotlivými vnějšími částmi a prvky, kterými je robot PR2 vybaven. Robot je poskládán z jednotlivých kloubů, spojů, rámů, převodů a pohonů. Tyto prvky jsou navzájem spojeny a tvoří tak větší celky, které jsou zpravidla upevněny na trupu robota. Za nejdůležitější části robota lze považovat základnu robota, teleskopický trup, robotická ramena a hlavu. Robot PR2 a popis jeho vnější struktury je zobrazen na obrázku 3.1.

- **Základna**
  Nejdůležitější částí celého robota je jeho základna. Základnu robota si můžeme představit jako schránku, v jejíchž útrobách jsou umístěny počítače, které jsou mozkem celého zařízení. Dále jsou zde uloženy baterie, které umožňují chod těchto počítačů bez

[1] Název odvozen z anglického Perosnal Robot.

připojeného napájecího kabelu. Základna obsahuje taktéž zadní panel, na který jsou vyvedeny jednotlivé porty, kterými lze k robotovi připojit jeho periferie. Na spodní straně základny je dále umístěna série kol, které jsou propojeny s motory, a které umožňují pohyb celého zařízení.

- **Teleskopický trup**
  Nejdominantnější částí robota PR2 je jeho trup, který je umístěn na pojízdné základně. Na trupu jsou po obou stranách připevněna dvě robotická ramena. V horní části trupu je umístěna hlava robota. Uvnitř trupu se dále nachází reproduktor a IMU jednotka. IMU jednotka je zařízení obsahující různá čidla jako například triaxiální akcelerometr, gyroskop nebo magnetometr. Trup robota je schopen se pohybovat ve vertikálním směru, přičemž rozsah tohoto pohybu je přibližně 30 cm.

- **Hlava robota**
  Hlava robota je s trupem spojena pomocí pohyblivého kloubu, díky kterému je možné s touho hlavou pohybovat nezávisle na jeho těle. Hlava je dále osazena sérií stereo kamer a jednou 5Mpx kamerou. Tyto kamery snímají prostředí, ve kterém se robot nachází. Součástí hlavy je dále projektor textur, který se používá k projekci textur na různé povrchy. Na základě této projekce je dále možné sestavit tří-dimenzionální strukturu povrchu, na který byla textura promítána. Dalším zařízením, které je na hlavě robota osazeno, je Hokuy laserový dálkoměr sloužící k mapování okolí.

- **Robotická ramena**
  Poslední hlavní součástí robota je jeden pár robotických ramen zakončených dvouprstými chapadly. Robotická ramena jsou k trupu robota připevněna pomocí kloubů, které umožňují vertikální pohyb těchto ramen. Na ramenech se dále nachází další soustava kloubů, které umožňují buďto celé rameno, nebo jen jeho část otáčet. Každé rameno je plně pohyblivé (lze mu přiřadit stupeň volnosti[2] číslo 7). Každé rameno je navíc opatřeno videokamerou, která zlepšuje odhad a ulehčuje případné uchopování předmětů. Dvouprstá chapadla jsou dále opatřena akcelerometrem, senzorem na měření stisku chapadel a kalibrační LED diodou.

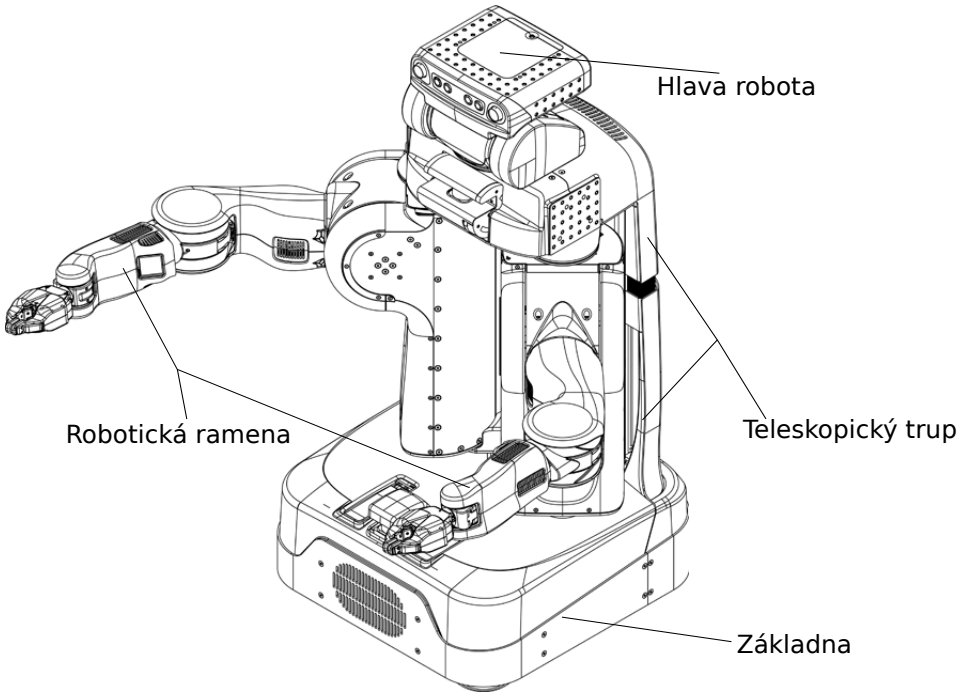
3.1.2 Vnitřní struktura robota PR2

Tato podkapitola je zaměřena na popis jednotlivých výpočetních zařízení (počítačů) potřebných pro správný chod robota. PR2 v sobě fyzicky obsahuje dva počítače (Počítač 1 a Počítač 2). K obsluze celého robota je potřeba však ještě jeden počítač označovaný jako Základní stanice. Základní stanice není fyzicky umístěna na robotu PR2. Hlavním cílem této stanice je zjednodušit vzdálenou komunikaci s Počítačem 1, která je realizována prostřednictvím bezdrátové sítě.

- **Počítač 1**
  Tento počítač je umístěn fyzicky na pravé straně základny a je označován za hlavní

[2] „Stupeň volnosti (angl. degree of freedom - DOF) je výraz používaný v robotice pro označení pohyblivosti. Aby těleso bylo plně pohyblivé (i když jen v omezeném rozsahu) musí mít šest stupňů volnosti. Tři z nich připadají na posun a tři na rotaci: Posun v ose x, posun v ose y, posun v ose z, rotace v ose x, rotace v ose y, rotace v ose z.“ [14]

[3] Obrázek byl převzat ze stránky http://www.cnet.com/news/getting-robots-to-do-the-laundry-and-the-dishes



Obrázek 3.1: Obrázek popisující robota PR2.[3]

počítač, jelikož se na jeho pevném disku nachází operační systém obou počítačů. Počítač 2 tedy nelze spustit, pokud není v provozu Počítač 1. Tento počítač je dále jako jediný přímo připojen k síti EtherCAT, která umožňuje ovládání motoru robota. V neposlední řadě Počítač 1 poskytuje směrování pro zbytek robota, ať už je robot připojen přes WAN port nebo přes jinou síť přes VPN tunel.

- **Počítač 2**
  Tento počítač je umístěn na levé straně základny a je označován za vedlejší počítač. Jednotlivé počítače jsou navzájem síťově propojeny.

3.2 Platforma ROS

ROS[4] je framework určený pro psaní a vývoj programů pro roboty. Jedná se o kolekci nástrojů a knihoven, jejichž primárním účelem je vytvořit komplexní a robustní prostředí použitelné na široké škále různých robotických zařízení a platforem. Pro dosažení a splnění těchto cílů musí ROS poskytovat hardwarovou abstrakci, ovladače zařízením, knihovny, vizualizéry, správu balíčků a mnoho dalšího. Jedná se tedy o komplexní systém, který lze chápat jako soubor procesů (uzlů). Tyto procesy lze libovolně seskupovat do různých balíčků (packages), které jsou lehce distribuovatelné. Další výhodou této platformy je její dostupnost. ROS poskytuje vývojářům možnost implementovat aplikace v různých programovacích jazycích jako jsou například Python, C++ nebo Lisp. Díky těmto vlastnostem a veřejné BSD licenci je ROS velice rozšířen a je s oblibou využíván výzkumnými skupinami zabývající se robotikou po celém světě [13].

[4]Robot Operating System

Koncepci ROSu lze rozdělit do tří základních úrovní. V následující části budou jednotlivé úrovně představeny a popsány. Dále jsou zde uvedeny a vysvětleny pojmy a termíny popisující vnitřní strukturu ROSu. Při tvorbě popisu úrovní jsem čerpal z oficiální dokumentace ROSu [4].

Výpočetní úroveň

Výpočetní úroveň se skládá ze sítě peer-to-peer procesů, jejichž primárním úkolem je zpracování dat. Základními prvky této sítě jsou:

- **Uzly (Nodes)**
  V ROSu uzly představují jednotlivé procesy. V každém zařízení je typicky spuštěno několik různých procesů, přičemž každý z těchto procesů spravuje a komunikuje s různými zařízeními, od kterých získává data, která jsou dále zpracovávána.

- **Kontrola řízení (Master)**
  Kontrola řízení provádí správu jednotlivých uzlů, registruje jejich názvy a zajišťuje vzájemnou viditelnost. Bez kontroly řízení by se do značné míry zkomplikovala komunikace mezi procesy a využívání jednotlivých služeb.

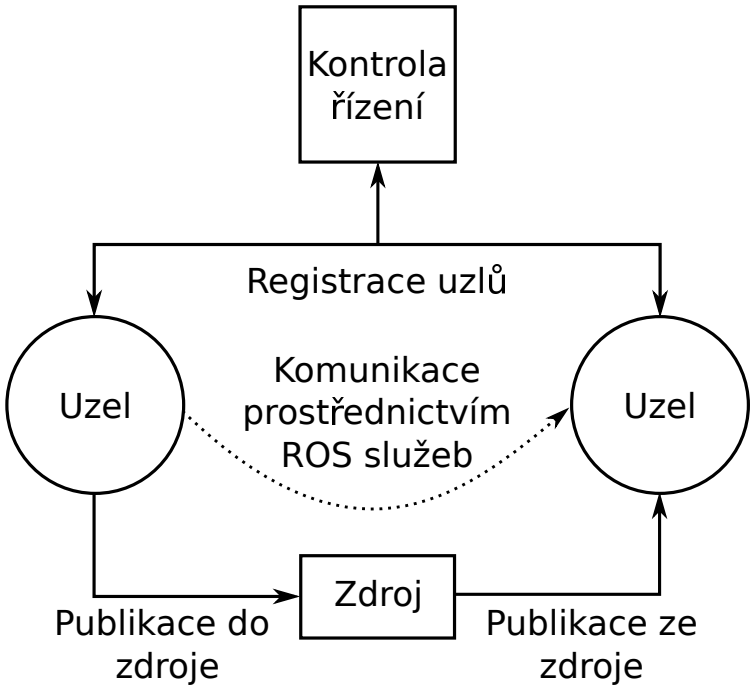
- **Parameter Server**
  Parametr Server je součást kontroly řízení a umožňuje datům, aby byla uložena podle klíče na centrálním úložišti.

- **Zprávy (Messages)**
  Jednotlivé uzly mezi sebou komunikují prostřednictvím zpráv. Zpráva je datová struktura obsahující položky. Tyto položky mohou mít jak standardní primitivní datové typy (celé číslo, číslo s plovoucí desetinnou čárkou nebo logickou hodnotu), tak i datové typy složené (pole). Zprávy mohou obsahovat libovolně vnořené struktury a pole.

- **Zdroje (Topics)**
  Zdroje zprostředkovávají komunikaci mezi konkrétními uzly. Uzel posílá zprávy na předem určený, pojmenovaný zdroj. Pokud má nějaký jiný uzel o tyto zprávy zájem, jednoduše se k tomuto zdroji připojí a posílané zprávy přijímá. Z jednoho zdroje může najednou odebírat zprávy více uzlů, přičemž jednotlivé uzly o sobě nemají navzájem povědomí.

- **Služby (Services)**
  Jelikož komunikace prostřednictvím zdrojů umožňuje pouze jednosměrný přenos dat, byly vytvořeny služby, které jsou založeny na principu žádosti a odpovědi. Služby jsou definovány pomocí dvojice zpráv (jedna zpráva pro žádost a druhá pro odpověď). Uzel tedy poskytuje konkrétní pojmenovanou službu, která čeká na příchod požadavku od klienta. Jakmile je tento požadavek přijat, dochází k jeho vyhodnocení a k odpovědi. Na obrázku 3.2 je ilustrován základní princip komunikace mezi jednotlivými uzly.

- **Bags**
  Bag je speciální formát sloužící k ukládání a následnému přehrávání obsahu zpráv. Typicky se takto ukládají data ze senzorů nebo kamer.



Obrázek 3.2: Komunikace mezi jednotlivými uzly.

Úroveň systému souborů

Do této úrovně lze přiřadit prostředky, které jsou fyzicky uloženy na pevném disku počítače. Typicky jsou to:

- **Balíčky (Packages)**
  Balíček je základní jednotkou pro organizaci programů v ROSu. Balíček může obsahovat procesy, knihovny nebo datové a konfigurační soubory. Lze ho tedy považovat za největší celek, který může být ROSem vytvořen.
- **Meta balíčky (Metapackages)**
  Meta balíčky představují speciální druh balíčků, které sdružují navzájem související balíčky.
- **Manifesty (Manifests)**
  Manifest v ROSu reprezentuje xml soubor, který v sobě obsahuje meta informace o balíčku, ve kterém se nachází. V manifestu jsou obsaženy informace o názvu, verzi nebo licenci. Jsou zde typicky uvedeny i závislosti balíčku na jiných balíčcích nebo stručný popis.
- **Repozitáře (Repositories)**
  Repozitář je sbírka balíčků, které sdílejí společný systém pro správu verzí (VCS system). Balíčky sdílející stejný VCS system mohou být sestavovány pomocí nástroje catkin. Catkin je tedy základní překládací systém ROSu, který kombinuje různá CMake makra a Pythonovské skripty. Repozitáře jsou často převáděny do kolekcí balíčků (rosbuild Stacks). Hlavním úkolem těchto balíčků je pak usnadnit jejich znovupoužitelnost a sdílení.

- **Typy zpráv (Message types)**
  Typy zpráv definují formát a datovou strukturu posílaných zpráv v ROSu.
- **Typy služeb (Service types)**
  Typy služeb definují formát a datovou strukturu požadavků a odpovědí na patřičnou službu poskytovanou ROSem.

Společenská úroveň

Tato úroveň poskytuje odděleným komunitám prostor, kde si mohou vyměňovat své nápady, programy a znalosti. Patří zde například repozitáře obsahující programy a aktualizace na roboty, wiki stránky obsahující dokumentaci k ROSu nebo různé blogy, kde si vývojáři pomáhají zodpovídají konkrétní dotazy.

Kapitola 4

Návrh a implementace

V následující kapitole je popsán princip funkčnosti mnou navržené aplikace, která vychází z výše uvedených algoritmů pro detekci a rozpoznávání obličejů v obraze. Jsou zde zmíněny problémy, se kterými jsem se při implementaci musel potýkat a které jsem musel vyřešit. Dále jsou zde popsány jednotlivé parametry, kterými lze do jisté míry zvýšit úspěšnost rozpoznávání a detekce v závislosti na okolním prostředí. V závěru kapitoly jsou uvedena omezení, která je nutno zohledňovat pro správný chod aplikace.

4.1 Obecný popis aplikace

Hlavním cílem této bakalářské práce bylo vytvořit aplikaci spustitelnou na robotu PR2, která by byla schopna detekovat, naučit se a následně rozpoznávat jednotlivé lidské obličeje nacházející se ve videu pořizovaném prostřednictvím kamer, které se na tomto robotu nacházejí. Jelikož se tedy jedná o robotickou aplikaci, celková koncepce této aplikace musela být navržena tak, aby zapadala do struktury platformy ROS, která byla popsána v kapitole 3.2. Jelikož ROS podporuje primárně programovací jazyky Python a C++, snažil jsem se implementační jazyk aplikace vybírat především podle jejich rychlosti (vzhledem k tomu, že aplikace je navržena tak, aby neustále prohledávala snímaný prostor a vyhodnocovala objekty a tvary nacházející se v tomto obraze, zvolil jsem jako implementační jazyk jazyk C++). Pro práci s obrazem jsem využil knihovny OpenCV, která je již integrována do platformy ROS. Pro detekci obličejů jsem použil algoritmus Haarových kaskád, jehož princip je vysvětlen v kapitole 2.2.1. Pro rozpoznávání detekovaných tváří jsem rovněž čerpal z prostředků knihovny OpenCV. Při volbě rozpoznávacího algoritmu jsem vybíral mezi algoritmem fisherfaces a algoritmem local binary patterns histograms. Po analýze a nastudování jednotlivých algoritmů jsem nakonec použil local binary patterns histograms. Princip algoritmu a jeho výhod je uveden v kapitole 2.2.2.

4.2 Funkcionalita a implementace programu

Program implementovaný v rámci mé bakalářské práce může být spouštěn ve dvou základních režimech činnosti. Prvním z těchto režimů je režim učení a rozpoznávání, druhým je pak pouze režim rozpoznávání. Implementačně se tyto režimy od sebe nijak zásadně neliší a jejich jádro zůstává stejné. V režimu rozpoznávání je však vynecháno volání některých funkcí na generování databáze obličejů získávané z kamer robota. Režim rozpoznávání dále vychází z předpokladu, že již má předem připravené adresáře s jednotlivými obrázky obličejů.

Dále je pro chod tohoto režimu nutný soubor `train.txt`, který obsahuje cesty k snímkům tváří a jejich unikátní identifikační čísla v předem definovaném formátu, který bude popsán níže. Tyto soubory lze automaticky vygenerovat prostřednictvím fáze učení a rozpoznávání. Takto vytvořené soubory pak přetrvávají i po ukončení aplikace. Je tedy možné z jednoho videa vygenerovat snímky jednotlivých tváří, které se na tomto videu postupně nacházely, a pak prostřednictvím fáze rozpoznávání tuto databázi použít na jiná videa, ve kterých se naučené osoby vyskytují. Další možností jak výše zmíněné soubory vytvořit je ručně, kdy je potřeba manuálně vytvořit snímky tváří, které máme zájem rozpoznávat a uložit je do patřičných adresářů. Dále musíme ručně vytvořit soubor `train.txt` obsahující všechny výše zmíněné náležitosti. Dalším souborem, který však již pro chod režimu rozpoznávání není nezbytně nutný, je soubor `id_name.txt`. Tento soubor obsahuje pouze mapování jednotlivých identifikačních čísel na konkrétní jména naučených osob. Pokud tento soubor nevytvoříme, je místo jména naučené osoby zobrazováno pouze její identifikační číslo.

Popis implementační části jsem rozdělil na dva logické celky. V prvním celku popíši detailně princip jednotlivých implementovaných algoritmů aplikace, práci s obrazem a algoritmus učení. V druhém celku bude popsána struktura aplikace z pohledu platformy ROS. Budou zde tedy popsány jednotlivé uzly (nodes), zdroje (topics) a služby (services), které aplikace využívá a které jsou aplikací vytvářeny.

4.2.1 Princip a funkce aplikace

Po spuštění programu nejprve dochází ke kontrole vstupních parametrů, kterými lze ovlivnit chod celé aplikace. Parametry lze upravovat prostřednictvím souboru `face_rec.launch`, který je umístěn v adresáři `launch` v kořenové složce projektu. Jedná se o xml soubor, v němž lze měnit hodnoty atributů `value` u elementů s názvem `param`. Ukázka jednotlivých parametrů je viditelná ve výňatku kódu pocházejícího ze souboru `face_rec.launch` 4.1.

```
<remap from="image_in" to="/wide_stereo/right/image_rect_color" />
<param name="LEARN" type="bool" value="true" />
<param name="THRESHOLD" type="double" value="60.0" />
<param name="IMAGES" type="int" value="2" />
```

Listing 4.1: Uživatelem modifikovatelné parametry obsažené v souboru face_rec.launch.

Prostřednictvím parametru `LEARN` lze aplikaci přepínat z režimu učení a rozpoznávání (pro tento režim je hodnota atributu `value` rovna logické hodnotě `true`) pouze do režimu rozpoznávání (pro tento režim je hodnota atributu `value` rovna logické hodnotě `false`). Parametrem `THRESHOLD` lze ovlivnit, zda se při nejednoznačné identifikaci má aplikace pokusit přiřadit jméno identifikované osoby na základě největší podobnosti s databází, nebo zda má porovnávanou tvář vyhodnotit jako neznámou (čím je hodnota atributu `value` menší, tím více se aplikace snaží vyhodnocovat detekovanou tvář jako neznámou). Parametrem `IMAGES` lze ovlivnit, kolik snímků obličeje daného člověka bude uloženo v databázi. Po kontrole datových typů jednotlivých parametrů jsou tyto parametry předány aplikaci, která si jejich hodnoty uloží do proměnných. Dále se rozhoduje na základě zvoleného režimu, jaké operace budou prováděny.

Byl-li vybrán režim učení a rozpoznávání, nejprve dojde k inicializaci a vytvoření jednotlivých adresářů a souborů. Je zkontrolováno, zda existuje adresář `$HOME/.ros/Face_rec`. Pokud tento adresář existuje, je i se všemi soubory v něm obsaženými smazán a vytvořen znovu. Složka `$HOME/.ros/Face_rec` obsahuje veškeré aplikací generované soubory sloužící k rozpoznávání jednotlivých osob. Stromová struktura adresáře `$HOME/.ros/Face_rec` je

zobrazena na obrázku 4.1. V adresáři `$HOME/.ros/Face_rec` se nachází následující textové soubory:

- `train.txt`
  V tomto souboru se nacházejí cesty ke konkrétním snímkům tváří, které jsou uloženy v jednotlivých podsložkách složky `$HOME/.ros/Face_rec/data`. Jednotlivé snímky jsou navíc doplněny identifikačním číslem, které je unikátní pro danou osobu v rámci souboru. Toto identifikační číslo je rovno číslu, kterým je ukončen název složky, ve které se dané snímky nachází. Cesta snímku je od identifikátoru oddělena středníkem. Pokud bychom zde cestu k snímku neuvedli, ale snímek by byl uložen v patřičné složce, bude ignorován. Soubor `train.txt` pro adresářovou strukturu uvedenou na obrázku 4.1 bude tedy vypadat následovně:

  Face_rec/data/person_0/picture_1.pgm;0
  Face_rec/data/person_0/picture_2.pgm;0

  Face_rec/data/person_1/picture_1.pgm;1
  Face_rec/data/person_1/picture_2.pgm;1

  Pokud bychom chtěli vytvářet databázi snímků a soubor `train.txt` manuálně, musí být dodržen výše specifikovaný formát.

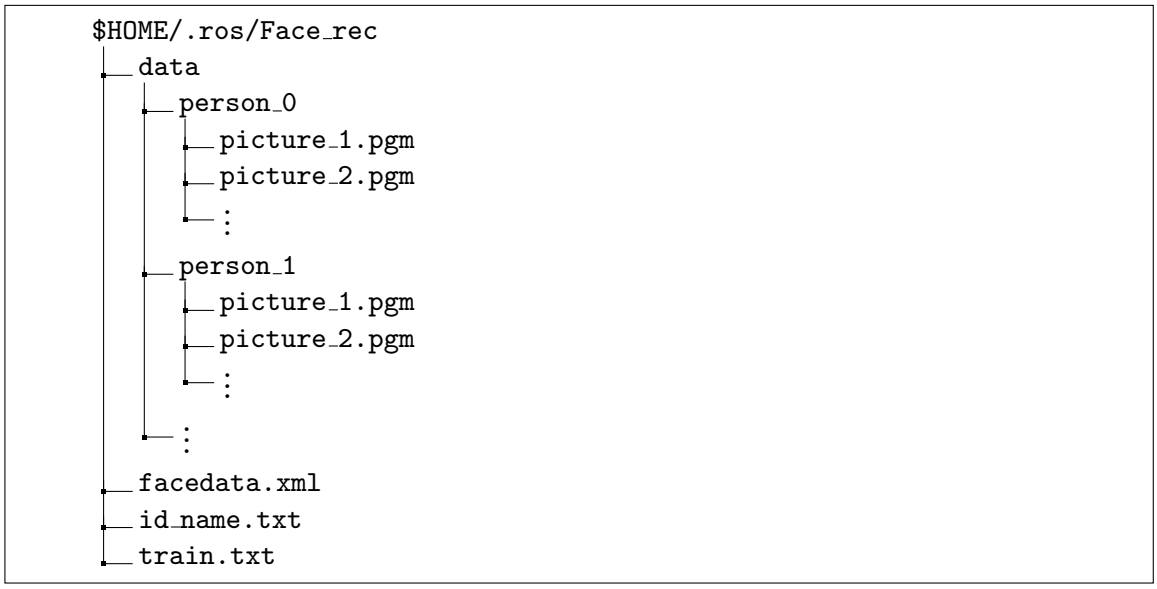
- `id_name.txt`
  V tomto souboru se nachází ke každému identifikačnímu číslu uvedenému v souboru `train.txt` textový popisek, typicky jméno patřičné osoby, kterým je pak ve výsledném videu obličej identifikován. Identifikační číslo je rovněž od textového popisku oddělováno středníkem. Při manuálním vytváření tohoto souboru musí být dodržen následující formát:

  0;xduban01
  1;xwrona00

  Pokud je soubor generován automaticky, je uživatel vyzván k zadání textového popisku k patřičné osobě. Zadávání se provádí pomocí ROS služeb. Podrobnější popis bude uveden v kapitole 4.2.2.

Mimo textových souborů se v adresáři `$HOME/.ros/Face_rec/data` nachází také soubor s názvem `facedata.xml`. Tento soubor je automaticky generován algoritmem local binary patterns histograms na základě snímků jednotlivých obličejů uložených ve složce `$HOME/.ros/Face_rec/data`. Jelikož se ve fázi učení a rozpoznávání může neustále zvětšovat počet osob nacházejících se v databázi, tento soubor je generován opakovaně pokaždé, když se shromáždí dostatečný počet snímků dané osoby specifikovaný parametrem `IMAGES`.

Pokud byl parametrem `LEARN` vybrán režim rozpoznávání, dochází pouze ke kontrole, zda existuje adresář `$HOME/.ros/Face_rec` a zda obsahuje všechny potřebné soubory v definovaném formátu. Po této kontrole jsou načteny jednotlivé snímky obličejů ze souboru `train.txt` a dochází k jejich naučení. Po dokončení tohoto procesu je vygenerován soubor `facedata.xml`, na jehož základě se provádí rozpoznávání v přehrávaném video souboru. Jelikož se v této fázi za běhu databáze tváří nemůže zvětšovat, je tento xml soubor vytvořen pouze jednou.



Obrázek 4.1: Adresářová struktura obsahující aplikací vygenerované soubory.

V následujících odstavcích nyní detailněji popíši princip vytváření databáze tváří u jednotlivých osob v režimu učení a rozpoznávání. Aplikace opakovaně analyzuje přijímaný obraz, ve kterém se prostřednictvím algoritmu Haarových kaskád vyhledávají lidské obličeje. Po detekci obličeje jsou souřadnice oblastí, ve kterých se tváře nachází, poslány k dalšímu zpracování a analýze. Existuje-li již v tomto okamžiku soubor `facedata.xml`, nejprve na základě informací v něm obsažných dochází k pokusu o identifikaci detekované tváře. Pokud se identifikace podaří a algoritmus local binary patterns histograms vyhodnotí detekovanou tvář za známou (tvář již prošla procesem učení), je této tváři přiřazeno patřičné identifikační číslo, popřípadě jméno, a výsledek je publikován do zdroje (topicu) `/final_image`, odkud je možnost výsledek detekce a rozpoznávání sledovat. Pokud však soubor `facedata.xml` neexistuje, nebo pokud identifikační algoritmus vyhodnotí detekovaný obličej jako neznámý, je zahájen proces učení.

Hlavním problémem, který jsem musel při fázi učení vyřešit byla identifikace jednotlivých tváří před možností použít k tomu určených algoritmů poskytovaných knihovnou OpenCV. Pokud je tedy v obraze detekována pouze jediná tvář, jednoduše je pořizován předem zadaný počet snímků této tváře přímo z obrazu videa. Dochází vlastně k ořezání detekované oblasti obsahující tvář, převodu tohoto snímku na šedotónový a k uložení do nově vytvořeného adresáře mající unikátní identifikační číslo. Jakmile je dosažen potřebný počet snímků dané osoby, je uživatel obsluhující základní stanici vyzván, aby rozhodl, zda si přeje později tohoto člověka identifikovat či nikoliv. Pokud se rozhodne, že si přeje pozdější identifikaci, je vytvořená databáze předána algoritmu local binary patterns histograms na naučení. Pokud však pozdější identifikace není požadována, je tato databáze smazána a tvář je nadále vyhodnocována jako neznámá. Spolu s tímto údajem je uživatelem zadáváno případné jméno osoby, která má být naučena. Tímto krokem je fáze učení ukončena a při opětovné detekci této tváře by docházelo pouze k její identifikaci. Toto je však pouze nejjednodušší případ. Problém nastává, pokud se v obraze nachází více tváří zároveň. V takovém případě výše popsaný postup ukládání detekovaných oblastí selhává. Docházelo by totiž k mísení snímků obou obličejů. Nejprve by se vytvořil a naplnil jeden adresář výřezy obou tváří. Po zadání vstupu by bylo učení u konce a následně by došlo k učení tváře

druhé. U této tváře by byly snímky již klasifikovány a zařazeny správně, jelikož by se v obraze nacházela již pouze jedna neznámá tvář. Tímto postupem bychom však nedosahovali korektních výsledků a docházelo by často k chybné identifikaci.

Pro eliminaci výše popsaného problému s mísením snímků jsem musel navrhnout algoritmus, který dokáže rozpoznat, kolik různých tváří se v obraze nachází. Jako nejsnazší řešení se mi jevilo určovat počet osob nacházejících se v obraze pomocí souřadnic jednotlivých tváří. Obdobný způsob jsem dále využil i při jejich identifikaci. Vycházel jsem z předpokladu, že lidé nacházející se současně v obraze mají mezi sebou určité rozestupy. Na základě toho jsem pak vyhodnocoval souřadnice získané detektorem. Každá souřadnice tedy reprezentuje konkrétního člověka nacházejícího se ve zpracovávaném obraze. Jelikož je potřeba si u každé souřadnice průběžně pamatovat poslední platnou pozici, počet aktuálně uložených snímků, unikátní identifikační číslo a cestu k příslušnému adresáři obsahující snímky stejných tváří, vytvořil jsem pole, ve kterém uchovávám jednotlivé struktury obsahující výše zmíněné položky. Obdrží-li příslušná funkce novou souřadnici neznámé tváře, nejprve dojde k prohledání všech doposud uložených souřadnic nacházejících se ve výše zmíněném poli struktur. Pokud v poli nejsou uloženy doposud žádné souřadnice, nebo pokud je aktuálně zpracovávaná souřadnice mimo poziční rozsah[1] uložených souřadnic, je algoritmem vyhodnoceno, že se jedná o novou tvář a je pro ni v poli vyhrazena položka (struktura), do které se uloží výše zmíněné potřebné informace. Pokud se však souřadnice nachází v pozičním rozsahu již uložené struktury pole, dochází k aktualizaci původní souřadnice obličeje, zvýšení počítadla uložených snímků a nastavení příznaku modifikovanosti, jenž je následně v další funkci kontrolován. Byl-li tento příznak nastaven, dojde k uložení snímku tváře nacházející se na aktuální zpracovávané souřadnici do patřičného adresáře. Dochází dále ke kontrole aktuálního počtu snímků jednotlivých tváří. Pokud je dosaženo maximálního zadaného počtu, je vytvořeno vlákno čekající na vstup od uživatele. Využití paralelismu je nezbytné, jelikož pokud bychom vlákna nepoužili, aplikace by se zastavila a čekala by, dokud uživatel nezadá vstup. Tím pádem by přestalo zpracovávání a publikování vstupního/výstupního videa. Po zadání vstupu uživatele dochází k vytvoření, naplnění a uložení souborů `train.txt` a `id_name.txt` a dále k inicializaci hodnot struktury, kterou daná tvář dočasně používala. Inicializací se tato struktura uvolnila, a může být tedy případně použita další příchozí tváří.

Jednotlivé detekované tváře jsou v obraze vyznačeny zeleným obdélníkem. Nad tímto obdélníkem se nachází identifikační číslo, popřípadě jméno identifikované osoby. Pro neznámé obličeje jsem použil jednotný identifikátor, jímž je číslo -1. Ve výsledném obraze, který obsahuje výše zmíněné značení, se může u detekované tváře objevit i obdélník červené barvy. Tímto značením je zvýrazněn obličej, u kterého aplikace očekává uživatelův vstup se jménem. V jeden okamžik však může být červeně označen pouze jedna tvář, aby nedocházelo k nejednoznačnostem. Konkrétní ukázka činnosti aplikace je zobrazena v příloze v kapitole C.

Algoritmus učení jsem po prvotním testování dále rozšířil o eliminaci chybných a neúplných detekcí. Tyto problémy byly způsobeny zpravidla dvěma důvody:

- **Chyba detektoru**
  V některých případech se stávalo, že detektor knihovny OpenCV poslal mnou implementované funkci na zpracování souřadnic pozici, na které se žádná tvář nenacházela. V takovémto případě došlo klasicky k zabrání jedné struktury pole a k vytvoření no-

[1] Pozičním rozsahem se rozumí rozmezí, jehož středem je uložená souřadnice, k níž je v kladném i záporném směru osy x přičtena konstanta určující celkovou velikost tohoto rozmezí.

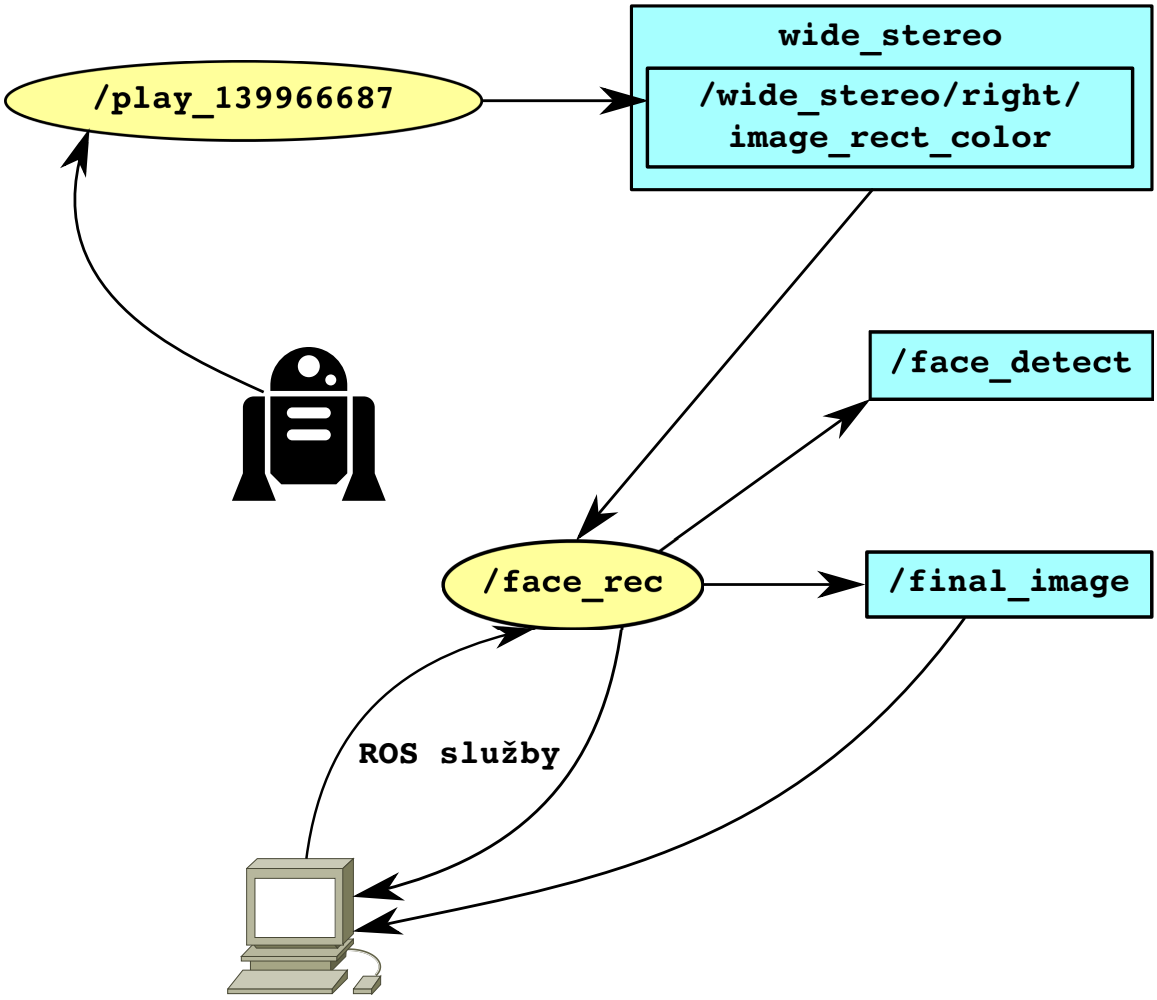
vého adresáře se snímkem z dané pozice. Problém však byl, že se daná struktura již nikdy neuvolnila, jelikož ona mylná detekce se neobjevovala pravidelně na stejných místech. Navíc se zbytečně vygeneroval adresář s jediným obrázkem, který nemohl být nikdy použit. Tento problém jsem vyřešil rozšířením podmínky, na níž záleželo, zda se provede uložení snímku obličeje do příslušné složky. Klasicky se zavolala funkce na zabrání struktury, kde se automaticky zvýšilo počítadlo celkových snímků jedné tváře. Toto počítadlo se však dále nezvyšovalo. Upravil jsem tedy podmínku, na níž záleželo, zda se zavolá funkce sloužící k vytvoření a uložení snímku. Tuto podmínku jsem rozšířil o logický výraz, který zajistí vyhodnocení celé podmínky jako pravdivé, pokud je počítadlo obrázku rovno pěti. Jinými slovy, obrázek tváře je uložen pouze tehdy, pokud je pět krát po sobě na stejných souřadnicích detekována tvář. Tato optimalizace však neumožňuje uvolnění zabrané struktury. Problém uvolňování zabraných struktur úzce souvisí s druhým problémem chybné detekce. Jeho řešení popíšu tedy tam.

- **Chyba způsobená nedostatečným časem na učení**
  Dalším problémem, se kterým jsem se při testování setkal bylo, že se aktuálně učená osoba nenacházela ve videu dostatečně dlouho. Vygenerovaly se například jen tři snímky z pěti požadovaných, a pak daná osoba opustila snímaný prostor. Vznikla tedy opět neúplná a nepoužitelná sada obrázků, která zbytečně zabírala místo na disku a která opět zabírala strukturu. Tato situace navíc mohla vést k mísení snímků dvou různých lidí. Toto by nastalo v případě, že by se na místo posledního výskytu první osoby postavila druhá, dosud neznámá postava. Za takových okolností by se pokračovalo v generování snímků do adresáře vytvořeného pro první osobu, což by mělo za následek špatné výsledky při rozpoznávání. Pro řešení právě popsaného problému jsem implementoval funkci, která pravidelně jednou za určitý počet zpracovaných souřadnic tváří kontroluje počítadla aktuálně uložených snímků v obsazených strukturách. Tento počet si pro jednotlivé obsazené struktury uchovává a porovnává je s hodnotami při dalším zavolání této funkce. Pokud jsou tyto hodnoty stejné, znamená to že na dané pozici se již žádná tvář nenachází a dochází k odstranění nekompletního adresáře daného člověka (pokud byl tento adresář vytvořen) a k uvolnění struktury pro další použití.

4.2.2 Vnitřní struktura aplikace z pohledu platformy ROS

Aby aplikace[2] byla schopna příjmat a následně zpracovávat obraz, musí se připojit ke zdroji videa (video topic), které je typicky nějakým uzlem (node) do tohoto zdroje publikováno. Obraz může být čerpán z řady zdrojů (v závislosti na kameře, ze které je obraz pořizován). Konkrétní zdroj je možné specifikovat v již zmíněném souboru `face_rec.launch` prostřednictvím atributu `to`, jenž je obsažen v elementu `remap` (viz. 4.1). Po přijetí se provede již popsaná analýza obrazu a akce s ní související. Výstupem je upravený přijímaný obraz obohacený o vyznačený obličej a textový popisek charakterizující jméno dané osoby. Takto upravený obraz je posílán speciálním typem zprávy (zpráva typu `image_transport`) do mnou vytvořeného zdroje nesoucí název `/final_image`. Na tento zdroj je typicky připojen uživatel obsluhující základní stanici, který publikovaný obraz sleduje na monitoru svého počítače a čeká, až ho aplikace vyzve k zadání jména právě naučené osoby. Uživatel

[2] V pojetí ROSu lze aplikaci chápat jako samostatný uzel, jenž komunikuje a posílá data do okolních zdrojů.



Obrázek 4.2: Komunikace mezi jednotlivými uzly (nodes) a zdroji (topics) aplikace. Žlutou barvou jsou znázorněny uzly, modrou zdroje.

interaguje s aplikací prostřednictvím ROS služeb (services). Jakmile uživatel zašle zprávu obsahující informace o jméně vyznačené osoby a o tom, zda má být databáze dané osoby předána algoritmu local binary patterns histograms k naučení. Poté aplikace zpětně informuje uživatele o zdaru nebo případném nezdaru provedené operace. Pokusí-li se uživatel zaslat zprávu aplikaci předtím, než je k tomu vyzván, dochází automaticky k neúspěchu. Celý výše popsaný postup je ilustrován na obrázku 4.2, kde samotná aplikace představuje uzel zvaný `/face_rec`.

Na obrázku 4.2 je uveden ještě jeden dosud nepopsaný zdroj, do kterého uzel `/face_rec` publikuje data. Je jím zdroj `/face_detect`, do něhož jsou posílány informace o známých nebo neznámých osobách v textové podobě. Jedná se o mnou vytvořený typ zprávy, jež obsahuje informace o souřadnicích, velikosti zobrazovaného obdélníku, který zvýrazňuje detekovanou tvář a jméno dané osoby. Konkrétní ukázka odchozích zpráv je viditelná na obrázku 4.3.

Informace publikované do tohoto zdroje používám primárně k vyhodnocování testů aplikace, kdy kontroluji, zda se v obraze nenachází jedna osoba se stejným identifikátorem vícekrát.

```
person: []
       ---
person:
  -
    x_offset: 173
    y_offset: 266
    height: 45
    width: 45
    name: -1
  -
    x_offset: 161
    y_offset: 256
    height: 68
    width: 68
    name: xduban01
---
```

Obrázek 4.3: Zprávy obsahující informace o obličejích nacházejících se ve videu. Položky struktury `person` *x_offset, y_offset, height* a *width* nesou informace o poloze a rozměrech rámečku ohraničující detekovaný obličej. Položka *name* obsahuje informaci o jméně detekovaného. Pokud se v obraze žádná tvář nevyskytuje, je struktura `person` prázdná (viz. první řádek obrázku).

4.3 Limitující faktory aplikace

Při testování aplikace jsem narazil na omezení, se kterými je nutné počítat, pokud chceme dosáhnout uspokojivých výsledků. Omezení plynou zejména z výše popsaného algoritmu učení, jenž pracuje s jednotlivými souřadnicemi tváří. Prvním omezením, na které jsem narazil pramenilo ze situace, kdy tvář absolvující proces učení příliš změnila svou polohu (dostala se mimo svůj poziční rozsah, který je jí vymezen). V takovém případě algoritmus vyhodnotí, že se v snímaném obraze objevila nová tvář a začne se jí učit od začátku (vytvoří jí nový adresář a ukládá do něj snímky). Pokud by tato osoba zůstala na nové pozici v rámci svého vymezeného pozičního rozsahu, nebo pokud by se opět vrátila na původní pozici tak by po chvíli došlo k uvolnění jedné ze struktur a smazání patřičného adresáře se snímky. Problém by však nastal, pokud by se daná osoba dostatečně rychle a pravidelně pohybovala z jednoho pozičního rozsahu do druhého. Tím by se průběžně generovaly snímky do obou adresářů. Jakmile by se nastřádal dostatek snímků v obou adresářích, byl by po uživateli vyžadován vstup pro jednu tvář dvakrát. Tento problém může z části vyřešit přímo uživatel zadáním stejného jména pro obě vyznačené oblasti. Dalším řešením problému, tentokrát na implementační úrovni, by bylo zvětšením pozičního rozsahu dané tváře.

Přílišným zvětšením pozičního rozsahu však nastává další problém. Čím větší je tento rozsah, tím méně jednotlivých tváří se může aplikace zároveň učit. Docházelo by totiž k překrývání pozičních rozsahů jednotlivých tváří. To by mělo za následek chybné ukládání do vytvořených složek (míchání obrázků vícero tváří) a tedy i pozdější nepřesnou identifikaci. Musel jsem tedy nalézt kompromis mezi velikostí pozičního rozsahu a počtem najednou se učících tváří. Nakonec jsem aplikaci upravil tak, aby bylo možno relativně bezchybně generovat databázi obličejů optimálně tří až čtyř různých osob současně.

Kapitola 5

Testování aplikace

V následující kapitole je uveden popis testů, na kterých se jsem testoval funkčnost a úspěšnost jednotlivých detekcí. Testování jsem prováděl na video souborech pořízených přímo z kamer robota PR2. Při testování jsem zkoumal závislost hodnoty prahu (parametr `THRESHOLD`) na celkovou procentuální úspěšnost rozpoznávání. Dále jsem v závislosti na tomto parametru pozoroval případy, kdy dochází při nerozhodné identifikaci k chybné záměně naučených tváří a kdy naopak je daná tvář vyhodnocena jako neznámá, i přes to, že se ve skutečnosti v databázi nachází. Dalším zkoumaným faktorem byl vliv velikosti databáze na úspěšnost detekcí. Při tomto pokusu jsem se snažil zjistit, je-li vhodným řešením mít vytvořenu jednu velkou databázi nejrůznějších osob, nebo zda je lepší při detekování mít uloženu menší databázi, ve které se nachází pouze osoby, které chceme identifikovat. V neposlední řadě jsem zkoumal, jak se změní procentuální úspěšnost detekcí v závislosti na počtu snímků jednoho člověka.

5.1 Popis testů a prezentace výsledků

Při testování aplikace jsem si nejprve prostřednictvím režimu učení a rozpoznávání vytvořil databázi konkrétních tváří, které se nacházely v pořízených záznamech. Takto vytvořenou databázi snímků jsem pak postupně používal pro analýzu jednotlivých videí, přičemž počet osob v databázi a počet snímků pro jednotlivou osobu jsem průběžně upravoval v závislosti konkrétním typu testu.

Při vyhodnocování testu jsem vycházel zejména z informací publikovaných do zdroje `/face_detect`. Procházel jsem tedy jednotlivé souřadnice a identifikátory příslušícím těmto souřadnicím, přičemž jsem se snažil najít chybné detekce. Mezi typické chyby patřila záměna jednotlivých identifikačních popisků, popřípadě vyhodnocení naučené osoby za neznámou. Tyto chyby však šly eliminovat nastavováním příslušných parametrů.

Počet snímků jedné osoby

V následujícím testu je zobrazen vztah procentuální úspěšnosti jednotlivých detekcí na počtu vygenerovaných snímků jedné osoby, který je uložen v databázi. Počet těchto snímků lze nastavit parametrem `IMAGES` v odpovídajícím souboru. Test jsem opakovaně prováděl na různých videích pokaždé s rozdílnou hodnotou prahu. V tomto testu databáze obsahovala pouze tolik různých osob, kolik se ve skutečnosti ve videu nacházelo. U všech videí jsem postupně nastavoval hodnotu prahu na hodnoty *50, 70* a *90*, přičemž jsem pozoroval vliv těchto hodnot na počet neúspěšných detekcí.

- **Video č. 1**
  V prvním typu testovaných videí se v obraze nacházely pouze dvě různé osoby. V prvním testu databáze obsahovala čtyři snímky jednoho člověka (celkem v databázi tedy osm snímků). Výsledky tohoto testu jsou zaznamenány v tabulce 5.1. V druhém testu jsem snížil počet obrázků jedné osoby na polovinu. Výsledky druhého testu jsou zobrazeny v tabulce 5.2. Sloupec tabulek se jménem *Počet detekcí* udává, kolik různých tváří bylo celkem v jednom videu detekováno.

  U tohoto testu jsem se mi podařilo dosáhnout nejvyšší procentuální úspěšnosti (*88 %*). Úspěšnost detekce se však nijak výrazně nezvýšila v závislosti na počtu snímků dané osoby. Větší rozdíly naopak byly patrné změnami hodnot prahu.

- **Video č. 2**
  U vyhodnocování druhého typu videa jsem postupoval obdobně. Zde se však již ve videu nacházely tři různé osoby, přičemž v prvním testu bylo v databázi obsaženo u každého člověka osm snímků jeho tváře. Výsledky tohoto testu jsou zobrazeny v tabulce 5.3. V následujících testech u tohoto videa jsem vždy snižoval počet uložených snímků u jednotlivých osob o polovinu. Výsledky pro databázi obsahující čtyři snímky u jedné osoby jsou zobrazeny v tabulce 5.4, pro databázi obsahující snímky pouze dva pak v tabulce 5.5.

  V tomto testu je již vliv počtu snímků v databázi u jednotlivých osob znatelný. Rozdíl úspěšností krajních případů (osm snímku a dva snímky) při stejné hodnotě prahu činí až *17 %*.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	85	32	30	2	62
70	85	10	0	10	88
90	85	10	0	10	88

Tabulka 5.1: Výsledky testu aplikovaného na videu č. 1. Každá osoba má v databázi čtyři snímky obličeje. Databáze sestává ze dvou různých tváří, přičemž všechny se ve videu objevují.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	85	40	40	0	53
70	85	14	2	12	84
90	85	12	0	12	86

Tabulka 5.2: Výsledky testu aplikovaného na videu č. 1. Každá osoba má v databázi dva snímky obličeje. Databáze sestává ze dvou různých tváří, přičemž všechny se ve videu objevují.

Hodnota prahu	Počet detekcí	Chybné detekce	Naučená tvář vyhodnocena jako neznámá	Záměna tváře	Úspěšnost detekce [%]
50	121	46	41	5	62
70	121	29	6	23	76
90	121	29	1	28	76

Tabulka 5.3: Výsledky testu aplikovaného na videu č. 2. Každá osoba má v databázi osm snímků obličeje. Databáze sestává ze tří různých tváří, přičemž všechny se ve videu objevují.

Hodnota prahu	Počet detekcí	Chybné detekce	Naučená tvář vyhodnocena jako neznámá	Záměna tváře	Úspěšnost detekce [%]
50	121	46	46	0	62
70	121	35	4	31	71
90	121	35	1	34	71

Tabulka 5.4: Výsledky testu aplikovaného na videu č. 2. Každá osoba má v databázi čtyři snímky obličeje. Databáze sestává ze tří různých tváří, přičemž všechny se ve videu objevují.

Hodnota prahu	Počet detekcí	Chybné detekce	Naučená tvář vyhodnocena jako neznámá	Záměna tváře	Úspěšnost detekce [%]
50	121	72	69	3	41
70	121	50	6	44	59
90	121	47	1	46	61

Tabulka 5.5: Výsledky testu aplikovaného na videu č. 2. Každá osoba má v databázi dva snímky obličeje. Databáze sestává ze tří různých tváří, přičemž všechny se ve videu objevují.

Velikost databáze

V následujících testech jsem přidal do databáze i snímky tváří, které se na přehrávaném videu neobjevovaly. Cílem bylo zjistit, jestli snímky tváře, které se v konkrétním videu nikdy nemusí objevit mají vliv na identifikaci osob, které naopak v záznamu figurují. Testy jsem opět prováděl s různou hodnotou prahu a s různým počtem jednotlivých snímků daných osob. Pro demonstraci vlivu velikosti databáze byly testy prováděny na stejných záznamech, jako předcházející test.

- **Video č. 1**
  První test se do značné míry podobá předchozímu testu u tohoto videa. Rozdíl je však v tom, že celková databáze kromě dvou lidí, kteří se ve videu nacházejí, obsahuje další čtyři, kteří zde však nefigurují. Výsledky tohoto testu jsou zobrazeny v tabulce 5.6.

  Při porovnání výsledků tohoto videa z předchozího testu (viz. tabulka 5.1) můžeme vidět, že rozšířená databáze nemá vliv na detekci. Dokonce jsou jednotlivé hodnoty a úspěšnosti úplně stejné.

- **Video č. 2**
  V tomto testu databáze obsahuje tři osoby, které se ve videu nacházejí a další tři osoby, které v obraze nevystupují. Opět zkoumám výsledky v závislosti na různé hodnotě zadaného prahu a na počtu snímků jednotlivých osob. Výsledky získané pro databázi obsahující osm snímků jedné osoby jsou zobrazeny v tabulce 5.8. Hodnoty získané pro počet snímků, který je roven čtyřem, jsou uvedeny v tabulce 5.9. Dvěma snímkům každé tváře pak odpovídá tabulka 5.10.

  Při porovnání jednotlivých tabulek výsledků z předešlého testu již ke shodným hodnotám úspěšných detekcí nedocházíme. Jsou zde viditelné nepatrné rozdíly, přičemž výsledky předešlého testu dopadly vždy o pár procent lépe. Nejmenšího rozdílu úspěšností jednotlivých testů bylo dosaženo při vyšším počtu snímků jednotlivých osob (rozdíl činil jen *2 %*). Znatelný pokles úspěšnosti však lze pozorovat u databáze obsahující pouze snímky dva. Zde byla hodnota rozdílu úspěšnosti jednotlivých testů rovna *12 %*.

- **Video č. 3**
  V dalším testu, který jsem prováděl měla aplikace k dispozici na naučení opět osm snímků každé tváře z databáze. Ve videu se nacházely pouze dva obličeje (zbylé čtyři se ve videu neobjevovaly). Databáze se tedy obsahovala snímky celkem šesti různých lidí. Konkrétní výsledky jsou zobrazeny v tabulce 5.11.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	85	32	30	2	62
70	85	10	0	10	88
90	85	10	0	10	88

Tabulka 5.6: Výsledky testu aplikovaného na videu č. 1. Každá osoba má v databázi čtyři snímky obličeje. Databáze sestává celkem ze šesti různých tváří, přičemž jen dvě tváře se ve videu objevují.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	85	40	40	0	53
70	85	14	2	12	84
90	85	12	0	12	86

Tabulka 5.7: Výsledky testu aplikovaného na videu č. 1. Každá osoba má v databázi dva snímky obličeje. Databáze sestává celkem ze šesti různých tváří, přičemž jen dvě tváře se ve videu objevují.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	121	47	41	6	62
70	121	31	5	26	74
90	121	31	1	30	74

Tabulka 5.8: Výsledky testu aplikovaného na videu č. 2. Každá osoba má v databázi osm snímků obličeje. Databáze sestává ze šesti různých tváří, přičemž tři tváře se ve videu objevují.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	121	49	46	3	60
70	121	39	5	34	68
90	121	39	1	38	68

Tabulka 5.9: Výsledky testu aplikovaného na videu č. 2. Každá osoba má v databázi čtyři snímky obličeje. Databáze sestává ze šesti různých tváří, přičemž tři tváře se ve videu objevují.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	121	78	75	3	36
70	121	63	9	54	48
90	121	62	2	60	49

Tabulka 5.10: Výsledky testu aplikovaného na videu č. 2. Každá osoba má v databázi dva snímky obličeje. Databáze sestává ze šesti různých tváří, přičemž tři tváře se ve videu objevují.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
50	393	353	333	20	10
70	461	171	24	147	63
90	413	136	1	135	67

Tabulka 5.11: Výsledky testu aplikovaného na videu č. 3. Každá osoba má v databázi osm snímků obličeje. Databáze sestává ze šesti různých tváří, přičemž pouze dvě tváře se ve videu objevují.

5.2 Vyhodnocení testů

Při testování aplikace jsem vypozoroval, že úspěšnost jednotlivých detekcí je silně závislá na parametru `THRESHOLD`. Optimální hodnota tohoto prahu pro detekci leží v intervalu od *80* do *100*. Záleží však na uživateli, jestli je pro něj přijatelnější častější záměna tváře, nebo vyhodnocování jako neznámá osoba. Nutno však podotknout, že pokud používáme aplikaci v režimu učení a rozpoznávání, musíme zvolit nižší hodnotu prahu. Toto je nezbytné, jelikož při vyšších hodnotách prahu často nedochází k vyhodnocování detekované tváří jako neznámé. Jelikož se však algoritmus učení spouští při detekování neznámé tváře, bylo by při nízkých výskytech těchto neznámých tváří učení problematické. Databáze, s níž jsem pracoval byla vygenerována s hodnotou prahu, která byla rovna číslu *60*.

Dalším faktorem od kterého se vyhodnocování odvíjí je počet snímků uložených v databázi u jednotlivých osob. Porovnávání provádím vždy na příkladech s nejvyšším a nejnižším počtem snímků s hodnotou prahu *90*. Porovnáme-li hodnoty úspěšností dosažených v rámci testů uvedených v tabulkách 5.3 a 5.5, kdy databáze obsahuje pouze tři různé osoby, vidíme zlepšení u identifikace s vyšším počtem snímků přibližně o *15 %*. Provedeme-li stejné porovnání výsledků testů s databází obsahující šest různých osob, dostaneme zlepšení až o *25 %*. Hodnoty jsou viditelné v tabulce 5.8 a 5.10. Vyšší počet snímků dále do jisté míry dokáže eliminovat chybu rozpoznávání, která vzniká v závislosti na velikosti databáze. Porovnáme-li například výsledek testu, kde se v databázi nachází pouze tři různí lidé (každý s osmi snímky) s testem, kde se naopak v databázi nachází šest různých lidí (opět každý s osmi uloženými snímky) dostaneme téměř nepatrný rozdíl úspěšností (*2 %*). Když si však porovnáme stejný případ, pokaždé však s databází obsahující dva snímky každé osoby, dostaneme rozdíl kolem *12 %*.

Nejlepších výsledků podle zobrazených tabulek bylo dosaženo v případě s vyšší hodnotou prahu a s vyšším počtem snímků dané osoby. Na základě těchto zjištění jsem se snažil zjistit průměrnou úspěšnost aplikace při jejím optimálním nastavení. Po analýze jsem zjistil, že průměrná úspěšnost aplikace se pohybuje kolem *70 %*. Podrobnosti jsou uvedeny v tabulce 5.12.

Hodnota prahu	**Počet detekcí**	**Chybné detekce**	**Naučená tvář vyhodnocena jako neznámá**	**Záměna tváře**	**Úspěšnost detekce [%]**
90	1309	401	8	393	69

Tabulka 5.12: Celkový přehled úspěšnosti detekcí.

Faktory ovlivňující úspěšnost detekcí

Při práci s vygenerovanou databází obličejů jsem narazil problémy, které do mohly ovlivnit výsledek jednotlivých detekcí. Těmito problémy jsou:

- **Velikost pořizovaných obrázků**
  Režim učení a rozpoznávání, jímž byla testovací databáze vytvořena, pořizuje snímky tváří přímo z videa, na němž se zároveň provádí jednotlivé detekce. Jelikož se však rozpoznávané osoby nacházejí mnohdy v relativně velké vzdálenosti od robota a jeho kamer, jsou pořizované snímky malé[1] a zachycená tvář na nich neobsahuje dostatek detailů.

[1] Největší pořízený snímek databáze měl velikost 68x68 bodů, nejmenší pak velikost byla 45x45 bodů.

- **Kvalita pořizovaných obrázků**
  Další nevýhodou plynoucí z pořizování snímků přímo z běžícího videa je pořizování rozmazaných obrázků. Často se při generování databáze stávalo, že snímek byl pořízen v okamžiku, kdy se daná osoba pohnula. V tomto případě opět obrázek neobsahoval dostatek detailů, což mohlo ovlivnit výsledky rozpoznávání.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo nastudovat metody detekcí a rozpoznávání lidských tváří v obraze. Na základě nastudovaných algoritmů a metod byl následně implementován program, který je schopen identifikovat různé obličeje.

Implementovaný program byl navržen pro robota PR2. K rozpoznávání jednotlivých tváří využívá aplikace algoritmus local binary patterns histograms, který pracuje s databází uložených snímků rozpoznávaných obličejů. Tuto databázi lze generovat přímo z videa na kterém se konkrétní tváře nacházejí. Po vygenerování databáze ji lze dále uložit a případně opětovně použít. Aplikace rovněž umožňuje nahrání vlastních snímků jednotlivých osob. Po spuštění aplikace doje pak k jejich naučení a následnému rozpoznávání v obraze.

Testování aplikace jsem prováděl na záznamech pořízených přímo z robota PR2, přičemž jsem zkoušel najít optimální nastavení aplikace, aby byla procentuální úspěšnost identifikací osob co nejvyšší. Při testování jsem analyzoval přes 1300 detekcí osob, přičemž jsem měnil hodnotu prahu, počet snímků dostupných u jednotlivých osob a velikost celkové databáze. Na základě těchto testů jsem určil optimální hodnotu prahu pro režim rozpoznávání na hodnotu 90. Pro tento optimální případ jsem dále znova analyzoval všechny detekce, přičemž zjistil, že z celkového počtu 1309 detekcí je 401 chybných. Průměrná úspěšnost aplikace při optimálně nastavené hodnotě prahu je tedy 69%.

Na tuto práci lze navázat například využitím informací o jednotlivých detekcích, které jsou posílány do speciálního zdroje. Aplikaci může být například rozšířena o možnost pohybu robota směrem k detekovanému člověku, popřípadě o možnost otáčení hlavy robota ve směru, výskytu lidské tváře. Další možností je využití vestavěného reproduktoru, kdy by byl robot na základě informací publikovaných do zdroje schopen oslovit naučeného člověka jménem.

Literatura

[1] Belhumeur, P. N.; Hespanha, P.; Kriegman, D. J.: *IEEE Trans. Pattern Analysis and Machine Intelligence*, ročník 19, 1997: s. 711–720, ISSN 0162-8828.

[2] Bradski, G.; Kaehler, A.: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 2008, ISBN 9780596516130, 580 s.

[3] Chang-yeon, J.: Face Detection using LBP features. *Final Project Report*, ročník 77, 2008.

[4] Foote, T.: ROS/Concepts. [online], 2013-12-23, [cit. 2014-05-08].
URL http://wiki.ros.org/ROS/Concepts

[5] Hlaváč, V.: Počítačové vidění vs. digitální zpracování obrazu, Digitální obraz a jeho vlastnosti. [online], [cit. 2014-04-24].
URL http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/Uvod+DigObraz.pdf

[6] Huang, T.: Computer Vision: Evolution And Promise. In *CERN.*, číslo č. 8 in Lecture Notes in Computer Science, European Organization for Nuclear Research, 1996, ISSN 0423-7765, str. 21.

[7] Licker, M.; Parker, S.: *McGraw-Hill Dictionary of Scientific and Technical Terms*. MCGRAW HILL DICTIONARY OF SCIENTIFIC AND TECHNICAL TERMS, McGraw-Hill Education, 2003, ISBN 9780070423138.

[8] Mašek, J.: Detekce objektů v obraze s pomocí Haarových příznaků. 2012.

[9] dev team, O.: Face Detection using Haar Cascades. [online], 2014-05-18, [cit. 2014-04-27].
URL http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

[10] dev team, O.: OpenCV 2.4.9.0 documentation. [online], 2014-05-19, [cit. 2014-05-19].
URL http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

[11] Wikipedia: Computer vision. [online], 2014-03-31, [cit. 2014-04-15].
URL http://en.wikipedia.org/wiki/Computer_vision

[12] Willow, G.: PR2 User Manual. [online], 2012-10-05, [cit. 2014-05-06].
URL http://pr2support.willowgarage.com/wiki/PR2%20Manual?action=AttachFile&do=view&target=pr2_manual_r321.pdf

[13] Woodall, W.: ROS/Introduction. [online], 2013-10-18, [cit. 2014-05-07].
URL http://wiki.ros.org/ROS/Introduction

[14] WP: Stupeň volnosti (robotika). [online], 2011-07-31, [cit. 2014-05-06].
URL http://www.i15.cz/stupen-volnosti-robotika/

[15] Šonka, M.; Hlaváč, V.: *Počítačové vidění*. Nestůjte za dveřmi, Grada, 1992, ISBN 8085424673, 252 s.

[16] Šťasta, R.: Podpora autodetekce obrázků pro systém G.A.T.E. 2012.

Příloha A

Obsah CD

Součástí mé bakalářské práce je jedno CD, které obsahuje zdrojové kódy mnou implementovaného programu a zdrojové kódy technické zprávy. Dále se na CD nachází video demonstrující funkčnost aplikace. Je rovněž přiložena databáze tváří, na základě které bylo video pořízeno. CD obsahuje následující složky:

- **Database**
  Adresář obsahující databázi obličejů a ukázku souborů, které jsou potřebné, pokud by byla aplikace z tohoto CD spouštěna rovnou v režimu rozpoznávání.
- **Doc**
  V této složce jsou obsaženy zdrojové texty tohoto dokumentu včetně všech použitých obrázků. Je zde přiložena i kopie tohoto dokumentu ve formátu PDF.
- **Src**
  Adresář se zdrojovými kódy samotné aplikace. V adresáři se nacházejí jednotlivé podložky obsahující potřebné soubory pro chod aplikace.
- **Videos**
  Zde jsou uloženy dva video soubory ve formátu AVI demonstrující funkčnost aplikace. Na jednom souboru je zachycen režimu učení a rozpoznávání, na druhém pak pouze režim rozpoznávání. Součástí je i BAG soubor, který představuje originální video z robota PR2.
- **Poster**
  V této složce je uložen ve formátu PDF plakát shrnující cíle a výsledky projektu.
- **Tests**
  V tomto adresáři jsou umístěny data získána ze zdroje `/face_detect`, které jsem průběžně ukládal při testování aplikace na různých videích. Na základě těchto dat jsem následně určoval úspěšnosti jednotlivých detekcí, které jsou popsány v kapitole 5.

Příloha B

Manuál

Následující kapitola obsahuje postup spouštění mnou implementovaného programu. Aplikaci lze spouštět přímo na robotovi PR2 nebo na osobním počítači, který ale musí mít nainstalovánu knihovnu ROS. Testování a spouštění aplikace jsem prováděl na operačním systému Ubuntu, pro nějž je platforma ROS primárně určena.

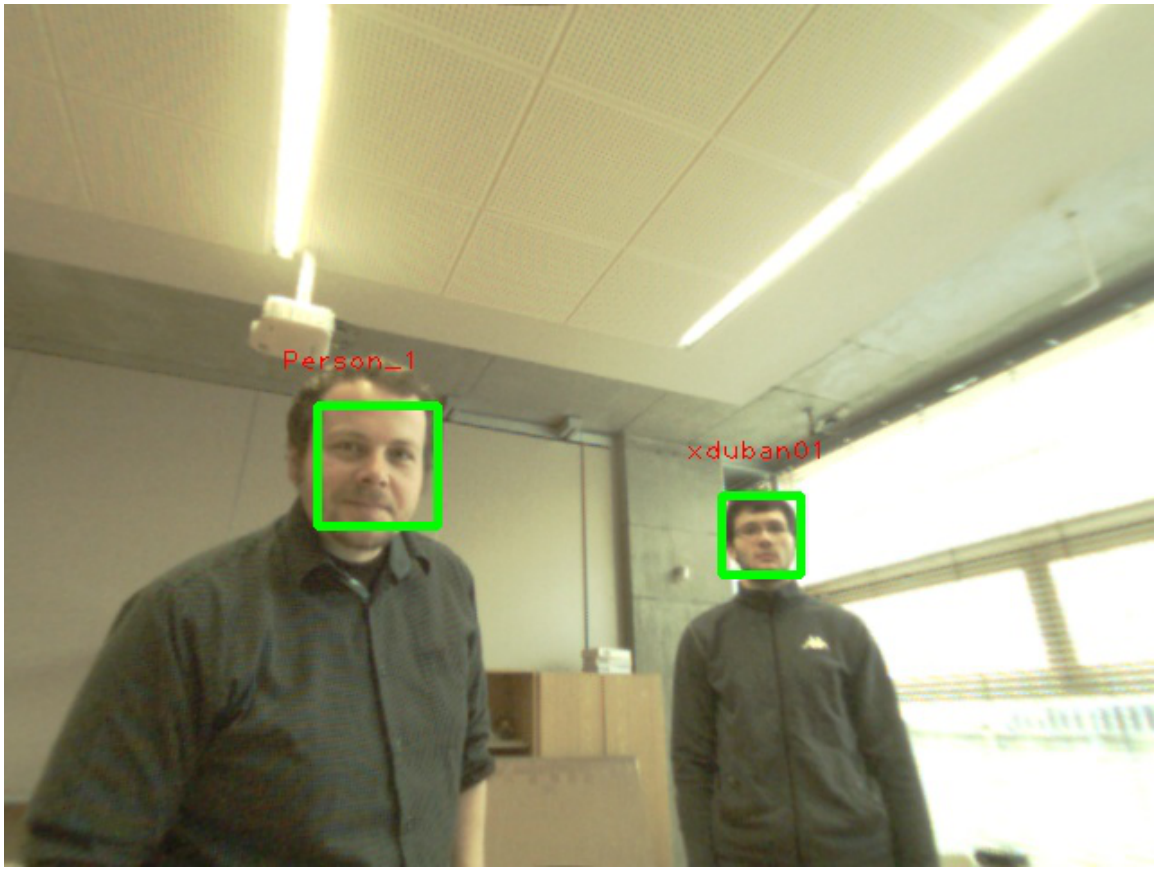
Pro přeložení programu je nutné si vytvořit na stanici pracovní adresář (*workspace* [1]), do něhož následně nakopírujeme obsah adresáře `Src`, jenž se nachází na přiloženém CD. Po nakopírování přejdeme do adresáře `Face_rec`, který obsahuje soubor `Makefile`. Překlad se zahájí zadáním příkazu `rosmake`. Před samotným spuštěním programu je možnost v složce `launch` upravit soubor `face_rec.launch`, který obsahuje nastavitelné parametry aplikace. Spuštění samotné aplikace se provádí příkazem `roslaunch Face_rec face_rec.launch`. Pro zobrazení výstupního videa, na němž jsou zobrazeny jednotlivé detekce je potřeba zadat příkaz `rosrun image_view image_view image:=/final_image`. Pokud byl vybrán režim učení a rozpoznávání, k interakci s aplikací dochází pomocí ROS služeb, které se spouštějí zadáním příkazu `rosrun Face_rec input_data` doplněným o příslušné parametry obsahující jméno naučené osoby a příznak určující, zda tvář má být naučena či nikoliv. Pokud je zvolen režim rozpoznávání, je nejprve nutné nakopírovat obsah složky `Database` nacházející se na CD do adresáře `$HOME/.ros`, ve kterém aplikace data očekává.

[1] Detailní postup vytváření pracovních adresářů z pohledu platformy ROS je uveden na stránkách http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

Příloha C

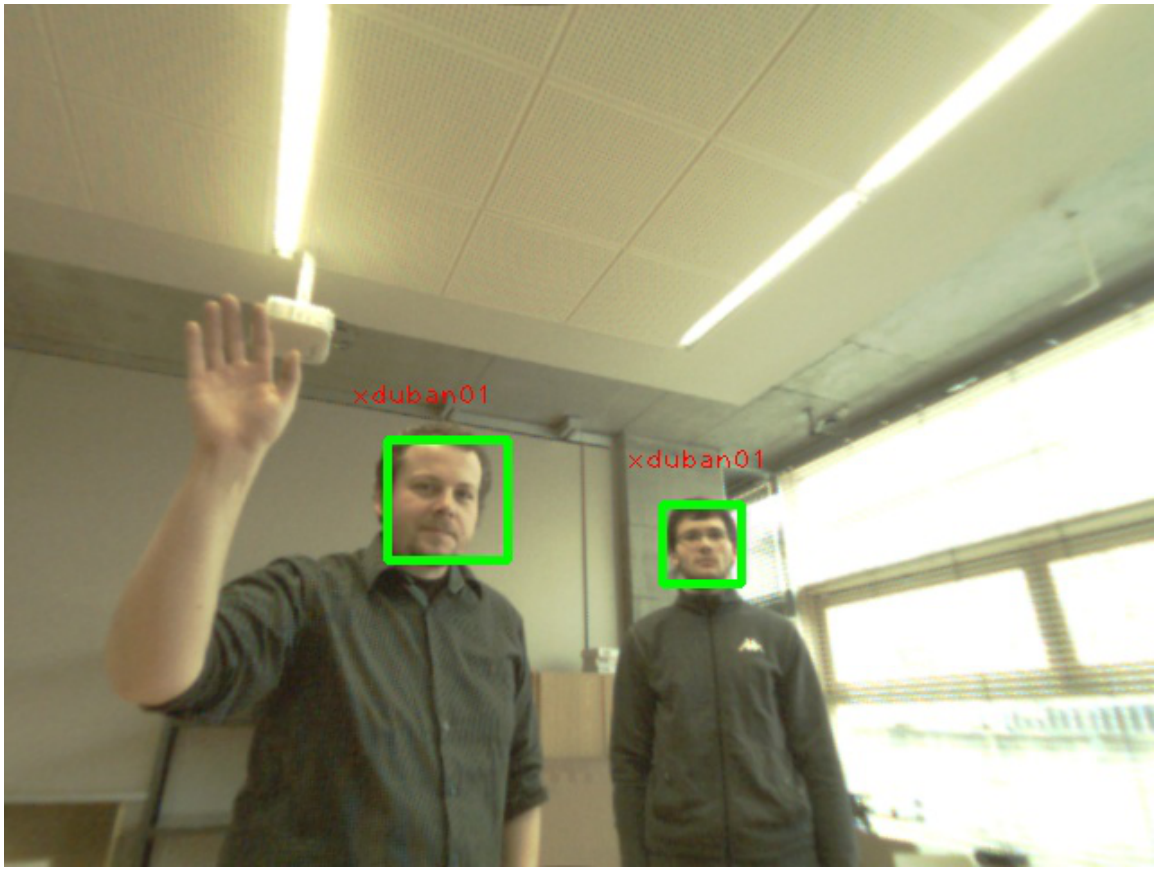
Ukázka činnosti aplikace

C.1 Korektní identifikace osob



Obrázek C.1: Obrázek zachycující korektní identifikaci tváří nacházejících se na videu.

C.2 Chybná identifikace osob



Obrázek C.2: Obrázek zachycující chybnou identifikaci tváří nacházejících se na videu.

Příloha D

Plakát

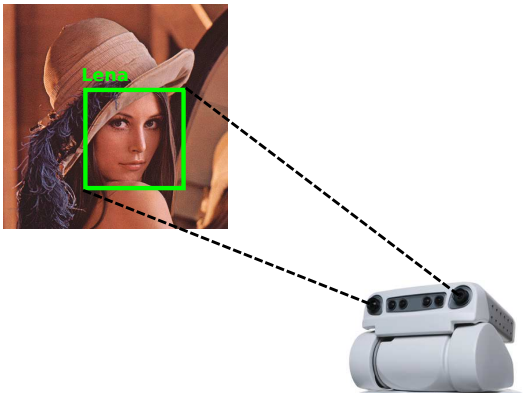
Rozpoznávání tváří

Cíle projektu:

- Detekce tváří v obraze
- Rozpoznávání tváří
- Robot PR2

Dosažené výsledky:

- Dodržena koncepce ROSu
- Generování databáze z videa
- Úspěšnost detekcí 70%



Použité algoritmy

- Local binary patterns histograms
- Algoritmus Haarových kaskád

Vliv prostředí:

- Kvalita snímků databáze
- Počet osob v databázi
- Vzdálenost od robota

Autor:
Michal Duban

Vedoucí:
doc. RNDr. Pavel Smrž, Ph.D.